UNIVERSITY OF OSLO
Department of Informatics

Comparison of Virtualization Performance: VMWare and KVM

Master Thesis

Naveed Yaqub

Network and System Administration

Oslo University College

May 23, 2012

Comparison of Virtualization Performance: VMWare and KVM

Master Thesis

Naveed Yaqub

Network and System Administration
Oslo University College

May 23, 2012

Abstract

The main purpose of this thesis is to compare the performance overhead of the virtualization infrastructures KVM and VMWare. All the experiments are carried out by using the Red Hat Enterprise Linux(RHEL) Operating System version 6.1. The study focuses on the performance of disk I/O operations, memory operations and CPU operations. The benchmarking tools used are Iozone for disk I/O, Ram Speed for memory and UnixBench for CPU.

First a set of benchmarking tests are carried out by using a Bare Metal installation of RHEL 6.1 on a Dell Poweredge R710 server. Next the exact same set of benchmark tests are run after installing RHEL 6.1 on a single virtual machine running on KVM on the same server. Finally VMWare ESXi 5.0 is installed on the server and RHEL 6.1 is installed on a single virtual VMWare machine. In this way the performance overhead of the two virtualization infrastructures KVM and VMWare is measured and compared. Each benchmarking test is run in each of the three cases sufficiently many times to produce statistically significant results.

The VMWare I/O disk performance is mostly from 20 to 30% better than KVM, with a few exceptions. And generally the VMWare I/O performance is 10-15% less than the Bare Metal performance. The memory performance overhead is relatively smaller. KVM performs better than VMWare for block sizes of 4MB and less, while the results show the opposite for block sizes larger than 4MB. When testing pure ALU usage, there is almost no virtualization overhead. There was some overhead for the other UnixBench CPU tests and in all these cases VMWare was performing better than KVM. Our general conclusion is that the virtualization overhead is less for VMWare than for KVM.

Acknowledgements

First, I would like to be grateful to my supervisor Hårek Haugerud for the project guidance and technical support. Thanks for him to make it possible for successful completion of this project. Report writing was a difficult challenge but was finished with his help. The arrangement of mid term presentation made me able to review my progress.

Secondly, I would like to thanks to Jarle Bjørgeengen as project provider at University Center for Information Technology at University of Oslo. I would like to appreciate him for his time to time guidance about project directions and special thanks for providing me hardware resource for project.

Thirdly, I would like to say thanks to Æleen Frisch for her valuable guidance on project report writing. I learned a lot from the lectures she delivered on report writing and it enabled me to proceed in the right direction. Next my gratitude is paid to Martin Kot for supplying me valid IP addresses and his technical support during the project. In addition, I am extremely grateful to my wife who assisted me in examining the thesis and corrected my English grammar mistakes.

Lastly I wish to take this opportunity to express my deepest thanks to my dearest parents, brothers and sisters for their unconditional love, support and encouragement throughout the study. I would also like to convey thanks to my lecturers, professors, classmates and friends. Their academic enlightenment, made my achievements possible during the two years master study.

Naveed Yaqub
May, 2012
Oslo, Norway

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Virtualization technology is considered the most demanding topic in todays era. Virtualization allows single computer to run multiples operating system simultaneously on a single computer system [1]. Virtualization technology helps the companies to run different services on a single server which enables to reduce the cost of managing more hardwares and usage of resources in more efficient ways. Now a days cloud computing is one of the most hot topic in computer system and virtualization is the key to the cloud computing [2].

At enterprise data center, virtualization technology makes it possible to minimize the costs by combining the server applications in fewer numbers of servers with reliable and secure way [17]. Different workload running on a single platform provides the better manageability, provisioning and cost [13]. Computer hardware is rapidly increasing its performance and thus tends to make some resources not to be fully utilized and virtualization technology is brought to overcome this problem [10]. Maximum utilization of computer system is become possible with the help of this technology.

There are many reasons to answer the question that why we need to use the virtualization as it has number of financial as well as managerial advantages. There are many challenges that can pop up while deploying the new applications and computer system specially today when modern hardwares are available for commercial and large scale enterprise use. The abstraction from the physical hardware is provided by the virtualization technology [3], that also removes the limitation of running only single operating system on a single hardware.

The concept of virtualization is not new and it started development as old as the beginning of the computer system. The pioneer of the virtualization technology was IBM when in 1960s and 1970s they introduced the technology in its System 360 and 370 mainframes. The PC based computer architectures invention during 1980s, the further development of virtualization was almost stopped [4]. The need for this technology was raised when ”people were waking up to the fact that the data center was full, the power they required had gone through the roof, and that they could not afford to continue to grow and scale out infrastructure,” [3] . Virtualization is now also supported by hardware manufacturer for example Intel and AMD, they extended the IA32 in-

struction set of x86 processor to support virtualization [5].

Because of plenty of great facilities offered by the virtualization, the market of virtualization technology is increasing rapidly and is creating interest for many companies. So to become vendors of virtualization tools such as VMWare, Xen, Red Hat etc. Lot of other system vendors such as IBM, Sun, and Microsoft are also now exploring virtualization technology.

VMWare is considered the market leader in virtualization technology with its strong product features. Some sources says that more then half of the virtualization market is captured by the VMWare and remaining half is shared by other vendors that includes Xen, Microsoft, Red Hat, IBM etc. Whereas the Red Hat claiming that after joining of KVM with red hat, they have now more secured and robust services in virtualization race. VMWare uses the virtual machine monitor VMM between operating system and hardware for management of the resources.

Dr. Rosenblum with his students at Stanford University launched several projects of virtualization technology. From this research group VMWare is created when Dr. Rosenblum with his colleague Diane Greene and two of his students. They started the company with the challenge to shape the product from this unproven research of university research group [2]. In 1999, VMWare introduced virtualization to x86 system for maximizing the system resources and transforms x86 system into a general purpose, shared hardware infrastructure that offers full isolation, mobility and operating system choice for application environments.

VMWare starts its journey when they are successfully allowing the access to applications for different operating systems. That success enables to test the performance of one application on different operating systems. After getting success in PC at commercial market, VMWare researchers started to think on new challenges of server virtualization. So that they could computing for storage and networking devices for building the datacenters [2].

A small company Qumranet an Israel based has developed KVM but in september 2008 Red Hat acquired Qumranet. KVM was ready for production at the time of acquisition by Red Hat. Red Hat realized that KVM virtualization technology is future based virtualization and now KVM is default virtual machine monitor in Red Hat Enterprise Linux (RHEL) since version 5.4 and the Red Hat Enterprise Virtualization for Servers [5].

1.1 Problem Statement

Virtualization is nowadays very hot topic. Most of the big market service providers are already enjoying the benefits of this technology. While the others are thinking to use it. As there are number of vendors available in the market and therefore, it needs to make detailed study about which technology is better than the other for a specific setting that

suits to an organization. This research work in general will study the virtualization technology with concentration of demanding market vendors that is VMWare and Red Hat. In more specific way the study will try to conduct the comparative research of virtual technology specifically overhead. That caused a little bit extra workload for processing to access the physical resources of the system. So the problem statement of this research work will be the:
Evaluate and compare the performance of VMWare and Red Hat KVM Virtualization.
It is more interesting to see the performance of Red Hat Enterprise Linux 6.1 as operating system. The performance comparison of these technologies shall be tested to use different benchmarking tools with respect to CPU, Disk I/O, and memory. This research work is primarily focused on comparison of Red Hat and VMWare virtualization technologies that needs to be addressed the following questions as well to be considered the part of this study:

- Performance of Red Hat Enterprise Linux 6.1 OS on Bare Metal (non-virtualized environment).
- Performance of Red Hat KVM virtualization with RHEL 6.1 OS installed on virtual machine.
- Performance of VMWare ESXi virtualization with RHEL 6.1 OS installed on virtual machine.

Detail review of the problem statement will further explain that there are three different scenarios that will be tested with three different performance measurement aspects. RHEL 6.1 operating system, Red Hat KVM, and VMWare. All these three different cases will be tested with three kinds of aspects that include CPU, memory, and disk I/O. To make the results comparable, identical environment will be provided by using the same OS that is RHEL 6.1 in all three cases. Each of these three aspects of performance will be repeatedly conducted for more reliable results that will be strong based for analysis and confident conclusion. Besides the primary questions that will be tried to search during this study, following secondary question will also be tried to answer with subject to availability of sufficient time.

- The scalability of Red Hat and VMWare virtualization while testing different components that is CPU, memory, and Disk I/O.

Detail study will be made to evaluate both the virtualization technologies with the help of their architectures and the way of their working. A middle layer in between hardware and operating system named hypervisor will be studied and discussed in detail that how it works. Hypervisor is an additional layer that makes a little delay for accessing the resources for virtualized environment and thus suffers the performance as compared with Bare Metal or non virtualized system.

Hence the study will address the performance comparison of two major cases that is virtualized and non virtualized environment and then two virtualization technologies. In first case where the comparison between RHEL 6.1 OS is made with Red Hat KVM and VMWare virtualized system. It is obvious that non virtualized system will perform better than virtual system as there is an additional layer exists in between virtual

operating system and hardware. This additional layer (hypervisor) has its own time delay that makes the performance of virtual operating system a little bit slower than operating system that is directly working with hardware. In second case where Red Hat KVM will be compared with VMWare, the results will be interested as both virtual technologies use different kind of hypervisor having their own architecture and have a different amount of overheads for system resources request of virtual operating system. Red Hat KVM virtualization as compared to VMWare is not that much mature as VMWare is considered as market leader for virtual technologies having biggest market stakeholder. Whereas Red Hat KVM virtualization having support in RHEL OS kernel is also part of red hat enterprise linux package.

A comprehensive analysis on the bases of data collection by using different benchmarking tools will enable to give our review and will suggest the best among Red Hat KVM and VMWare.

1.2 Objective and Methodology overview

This study is conducted to know the performance comparison of the virtual machine monitor. As there are many virtualization technologies available and it also known as that virtualization introduce some overhead that caused the performance suffering. As different virtualization technologies use different kind of virtual machine monitor or hypervisor so the performance is also varied [36]. The objective of this research work is to figure out the following questions:

- The performance degradation of virtual machine against physical machine.
- How much difference between different virtualization technologies?
- What factors lead to the performance loss of virtualization systems?

1.3 Research structure

Goal of this research work is to measure the performance overhead of virtualization technology with native system. In first chapter problem statement and objective are discuss after introducing the virtualization technology. Second chapter presents the history of virtualization and its different types. Benefits of virtualization and description of VMWare and KVM along with background material and different overhead is also the part of the second chapter. The third chapter will introduce the benchmarking and different available benchmarking tools. The specification of hardware and implementation of benchmarking tool will be presented in fourth chapter. Result and discussion chapter will show different graphs and analysis of few selective cases for discussion. This report will be concluded in chapter six with final words and future research question.

Chapter 2

Background and literature

2.1 Virtualization

There are many definitions of virtualization but the following are more general by [16]

Definition 1. *"Virtualization is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, timesharing, and many others."*

By this definition virtualization provides isolated environment over the hardware for application of operating system. As the users want to get maximum utilization of the hardware, so this virtualization technique provides the opportunity for optimum benefits from hardware resources that are closed to real machines. Another definition of system virtualization is given below [17]

Definition 2. *"A system VM provides a complete environment in which an operating system and many processes, possibly belonging to multiple users, can coexist."*

The system virtualization from above definition provides the usual hardware like ethernet controllers, CPUs or hard disk drives to an operating system which runs inside of it. Such a system that has attached physical hardware is capable of running many virtual machines at a time that is known as virtualization host whereas the virtual machines running on it are called guest. The operating system running on each virtual machine is known as guest operating system.

In virtualization systems, usage of hardware resources between parallel running of virtual machines are managed by special software known as Virtual Machine Monitor (VMM) or hypervisor that works between the hardware and operation system [14]. Hypervisor creates illusion of hardware resources to make it possible to run multiple virtual machines at the same time [15]. A key for every virtual technology is called hypervisor, that allows the system resources for each OS program[1]. The hypervisor basically works in between different OS and system resources. Multiple OS's are competing for resources such as CPU, memory, data, network etc. and hypervisor is responsible to manage all such requests.

2.1.1 History of Virtualization

Before proceeding to know about server virtualization technology, it is need to understand why and how this technology discovered. The first computing systems developed were large in size and expensive to operate. As the computer system got popular its demand was also increased and that firstly brought the idea of batch processing in 1950 which afterwards emerged into time sharing system in 1965. This time sharing system then allows multiple applications from number of users to run at the same time. Time sharing systems were working fine but the problem arose when error in one application crash the entire system. To increase the reliability of the system, the applications need to be isolated from each other.

In the beginning one system was used for one application for isolation purpose but this was very expensive [18]. It was not only expensive solution but also wastage of system resources as well as the system was utilized with their full capacity because of non time sharing system. This problem became a reason for development of new ideas and then software development introduced the instruction-by-instruction simulation of one computer system differ from the other system. This isolated application creates copies of hardware and software for each application that facilitates the user to run their own application even operating systems of their own that evolved the idea of virtual machine [19].

According to Marinescu [18] the isolation of the application by mean of virtual machine had the drawback of slowing down the computer system by 20 to 1 factor as it shares the hardware resources among different operating systems. For monitoring the performance of simulation softwares to improve the system efficiency, the idea of virtual machine monitor was introduced in 1960s and IBM virtual monitor/370 was considered one of the system at that time.

The virtualization technology was improved in 1970s and widely used by the organizations because of cost effectiveness. During the 1980s and 1990s the cost of computer hardware was dropped and multitasking operating systems were also not that much used. During that era virtualization technology was not given consideration for improvement. The one reason for not using the virtualization might be because of low prices of system hardware, maximum utilization of system resources were also not that much concern whereas before 1980s system architecture was developed for keeping in mind of virtualization [20]. The new computer system architecture from mainframes to minicomputers and then PCs did not have the support for virtual machine monitor [23].

VMWare was the first organization in 1990s, when virtual machine monitor was researched by one of the research group of Stanford University in a research project. VMWare became the pioneer of virtualization technology for commodity hardware. After VMWare successful reshaped of the virtualization technology, lot of other vendors has also shown their interest for making research and development in this field. Unlike in the beginning of virtualization technology when it was introduced for multitasking, nowadays it is used to decrease the management cost along with hardware cost. Now lot of vendors with variety of different alternatives are available in the mar-

ket that include SWsoft and XenSource. The hardware system architecture is also now virtualization supported for example Intel (VT) and AMD (SVM) to support virtualization [18].

2.1.2 Basic Concepts of virtualization

The software abstraction between hardware and operating system is called virtualization. All the applications are run in the operating system that is running over the abstraction layer also called virtual machine monitor or hypervisor [18]. The hypervisor is used to hide the hardware system resources from operating system that allows to run different operating systems at the same time as the hardware is not directly accessible by the operating system. Further Marinescu [18] explains that available hardware is logically divided into number of logical units that each called virtual machine.

Every virtual machines have a set of following requirements [20]:

- **Equivalence:** The application that is running in virtual machine is just like same as it is running on the hardware without any additional plugin requirement. It must be identical in behavior while running in two different case.
- **Control:** The abstraction layer in between hardware and virtual machines must be controlled and synchronized access of virtual machines to hardware resources.
- **Isolation:** Virtualization technology was developed to ensure the isolation in between virtual machines. The purpose is to ensure stability of crashing one virtual machine should not affect others. Security from compromised virtual machine should not grant access to other virtual machines and data consistency.
- **Performance:** The virtualization overhead that is due to abstraction layer should be minimal, almost close to "Bare Metal" performance.
- **Encapsulation:** Cloning of the virtual machines are easy when they exist in the form of directory of file that also allow easy migration of the virtual machines.

2.2 Classification of virtualization techniques

Virtualization can be classified into following three different categories for a particular intel x86 architecture [18]:

- Full Virtualization
- OS-Layer or Para Virtualization
- Hardware-Layer Virtualization

All these types are further explained [18]:

2.2.1 Full Virtualization

In full virtualization, virtual machine monitor is also known as virtual machine manager that runs over the host operating system just similar as user application. This type of virtualization provides virtual hardware that allows the virtual machines and guest operating system to run on top of virtual machine manager as shown in the figure 2.1.

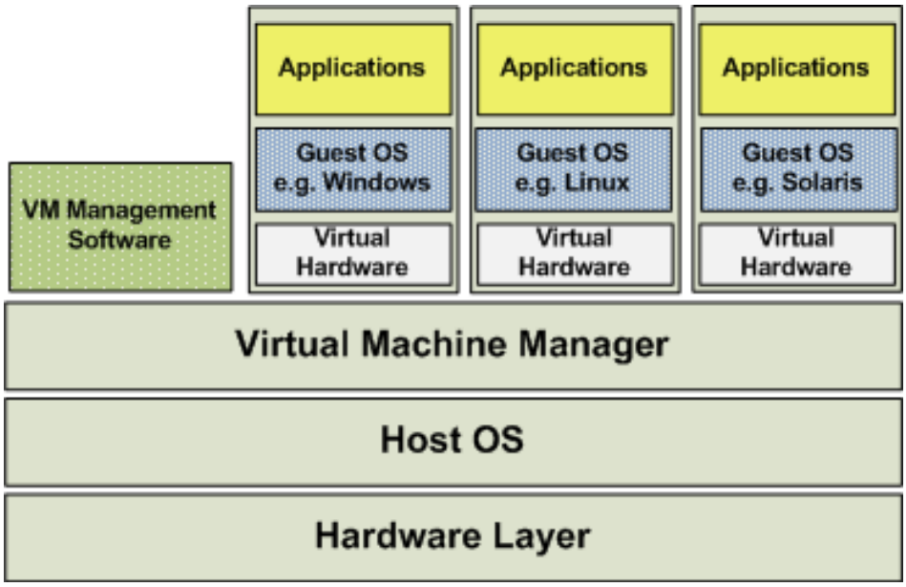

Figure 2.1: Full Virtualization [18]

Many of the market vendors including VMWare Workstation, Parallels, and Virtual PC are using full virtualization products, whereas VMWare workstation is the most popular among them. VMWare renamed the virtual machine manager as hosted virtual machine architecture that install VMWare workstation on top of the host operating system.

This type of virtualization is easy to use for layman as it only needs to install a software product like VMWare Workstation similar to any other software product on its operating system of users own choice. Guest operating system can be installed on VMWare workstation that will work as normal operating system using system hardware resources. According to Marinescu [18] the performance of the guest operating system can be decreased up to 30% because of additional host operating system and virtual machine manager in between hardware and guest operating system. In desktop market, users are more concerned with usability instead of performance, so the poor performance even is not main hurdle in success of this type of virtualization. But on the other hand in server technology, where performance is the main factor along with others, that arises the need of separate virtualization solution.

2.2.2 OS-Layer or Para Virtualization

Unlike full virtualization where virtual hardware is created, in operating system layer virtualization, it implements various instances of virtual guest operating system. The virtual machines use the virtualized image of operating system. This kind of virtualization is also known as Single Kernel Image (SKI) or container-based virtualization. Virtual server operating system is virtualized in this approach, which allows the hypervisor and the guest OS to collaborate on achieving the fastest, most robust performance possible [4]. The figure 2.2 explains the operating system layer virtualization architecture.

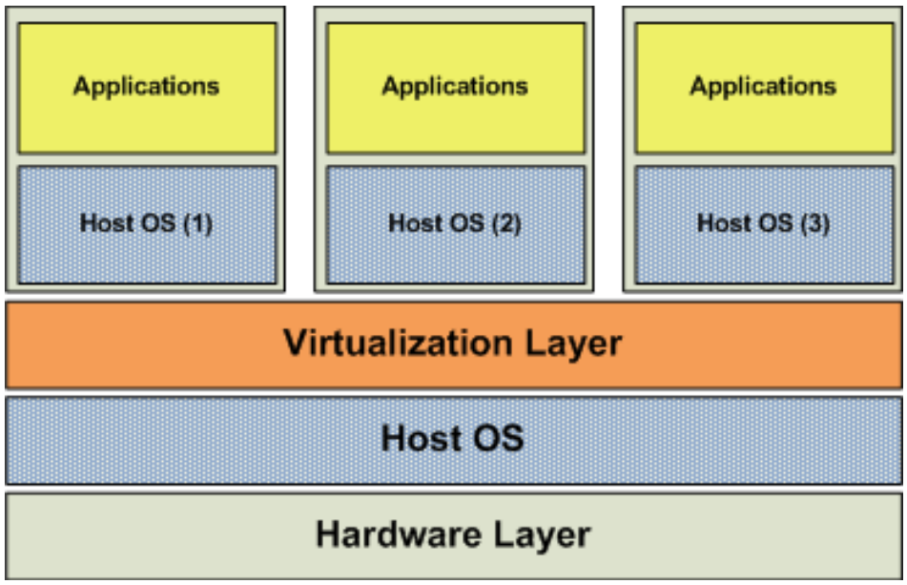

Figure 2.2: OS Layer Virtualization [18]

Each guests can run complete operating system in paravirtualization approach whereas the privileged instructions cannot be executed by a guest. Modification in guest operating system is required to implement an interface for running of privileged instructions on guest virtual machines. Virtual machine monitor can run the restricted instructions for virtual machine. This approach is closed to native performance but lacks in the support for closed source operating system [12]. The source code of kernel of an operating system is needed to be patched for mentioned modifications and this makes impossible for Microsoft Windows to run in a VM using paravirtualization.

Operating system layer virtualization is mostly used in web hosting, high performance computing (HPC) clusters and grid computing. This approach is used in different products that includes Virtuozzo and its open source variant OpenVZ, Solaris Container, BSD Jails and Linux VServer. System can easily manage and administrate with this approach, system resources can be assign to virtual machines at the time of creation and run time as well that includes memory, CPU guarantees and disk space [18]. This type of virtualization is more efficient as compared with other server virtualization

solutions and very few instances of failures to provide the isolation [22]. A big disadvantage of this type of virtualization is guest and host operating system must be same because the virtual machines are using the same kernel as the host operating system mean that means it is not possible to run e.g. Windows on top of Linux.

2.2.3 Hardware-Layer Virtualization

High performance and isolation is most commonly used by the hardware layer virtualization for server technology. In this category the virtual machine monitor directly runs on hardware that controls the access of the guest operating systems to the hardware resources as shown in figure 2.3:

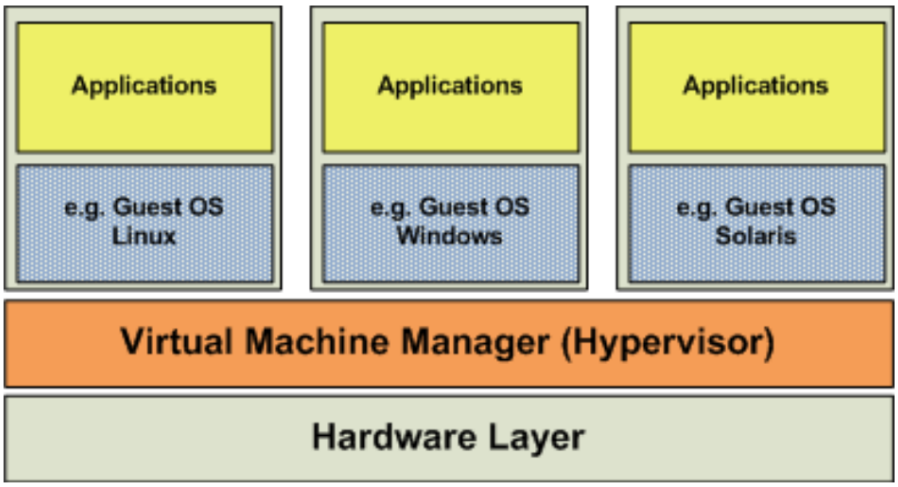

Figure 2.3: Hardware Layer Virtualization [18]

Market leaders in virtualization VMWare ESXi server and Xen are using this technology. According to Marinescu [18] "Paravirtualization is the technique used by Xen which provides a virtual machine interface representing a slightly different copy of the underlying hardware, where the nonvirtualizable portions of the x86 original instruction set are replaced with their easily virtualized equivalents". Rosenblum [23] further says that "guest operating systems running on top of this interface must be ported to use the slightly modified instruction set". At the same time running application on guest operating systems does not required any alteration. Porting an OS to Xen is relatively low in cost [24], but still its a big disadvantage of the paravirtualization approach. Adams and Agesen agree that "the approach used by VMWare ESX avoids this problem by performing on the fly binary translation of privileged code (kernel code) which replaces the nonvirtualizable instructions with a code that can be run directly on the CPU" [25].

2.3 Virtualization Usage Benefits

Virtualization technology has plenty of benefits for using. Some main advantages are listed below[7].

- Workload consolidation can be possible with the help of virtual machines to use the fewer machines, even on single server can be used. Virtualization for workload consolidation having a benefits of savings on hardware, environmental costs, management, and administration of the server infrastructure.
- Untrusted applications that are vulnerable for the system can be isolated by using separate virtual machines which are an important concept in building secure computing platforms.
- Execution environments with resource limited operating system can be created for a specific purpose. For example if an operating system that don't needed graphical environment or other resources like NIC etc can be created that might be able to increase the quality of service enabled operating system.
- As there is only one physical machine for number of virtual machines but all the virtual machines consider having their own hardware resources that are kind of illusion of hardware. Independent networks can also be simulated with the help of virtualization technologies.
- Simultaneously support for running multiple operating systems can also be possible by using he virtual machines. Even same operating system with different version or different operating systems that are some time difficult to run on real hardware is possible with virtualization environment.
- Virtual machine monitoring tool can be configured for debugging and performance measurement of the virtual machine.
- Software migration is easy with virtual machines that adds system mobility features.
- Research and academic experiments that might be risk for system crushing, virtual machines environment can be a great tools for them. Since they provide isolation, they are safer to work with.
- A test scenario can be created for an application that can be led towards implementation in real environment for effective quality assurance.
- A new feature of operating system can be tested on virtual machine before its implementation at actual.
- System backup, recovery, or migration is quite easy and manageable by using virtualization.

2.4 Server Virtualization Technology

When limited hardware resources are available for services or when optimum usage of resources are required for a server then server virtualization technology is handy to use, that allows to make multiple virtual servers on one physical server machine. An operating system for each virtual machine can be ran that is just shadow of a physical server. According to Oguchi and Yamamoto [6] Virtual machines can be built by dividing a physical server in terms of hardware and software. More they describe the advantages to this division of hardware that error in one section is not affecting the others. On the other hand software divide the system resources that include CPU, memory, I/O devices, and other hardware resources to be assigned to virtual machines, and the resource assignment status changed even during operation. Moreover, when the physical server is divided by software (through server virtualization), the hardware resources can be shared by and assigned among multiple virtual machines, as well as being used exclusively by a specific virtual machine.

Server virtualization starts when Dr. Mendel Rosenblum, associate professor of Computer Science at Stanford University, initiate his research group to focus on machine virtualization as a technique for utilizing servers with various system services [2].

The server virtualization technology is expected to reduce the construction costs of application servers that is nows largely used. Cost of expenses, required time, location and others is expected to decrease by implementing the server virtualization technology [6]. The maximum usage of hardware resources of the server is possible with server virtualization technology. As the server is expected to perform significantly well specially during peak rush hours, but possibly rest of the time the usage of hardware might remain low. To make sure the services availability to client a backup server is also usually maintained by the companies that caused to increase the system cost. Server virtualization technology can help to reduce such cost.

On a single server, when having multiple virtual machines are installed as shown in figure 2.4, the installation cost and time needed can be reduced as compared to a setting up a multiple physical servers. Using the server virtualization technology, it only required a single hardware server resources and decreases the expenses. Moreover, building multiple systems on one physical machine enables to maximum utilization of the resources e.g CPU, memory, I/O, and network interface will result in more effective use of server resources.

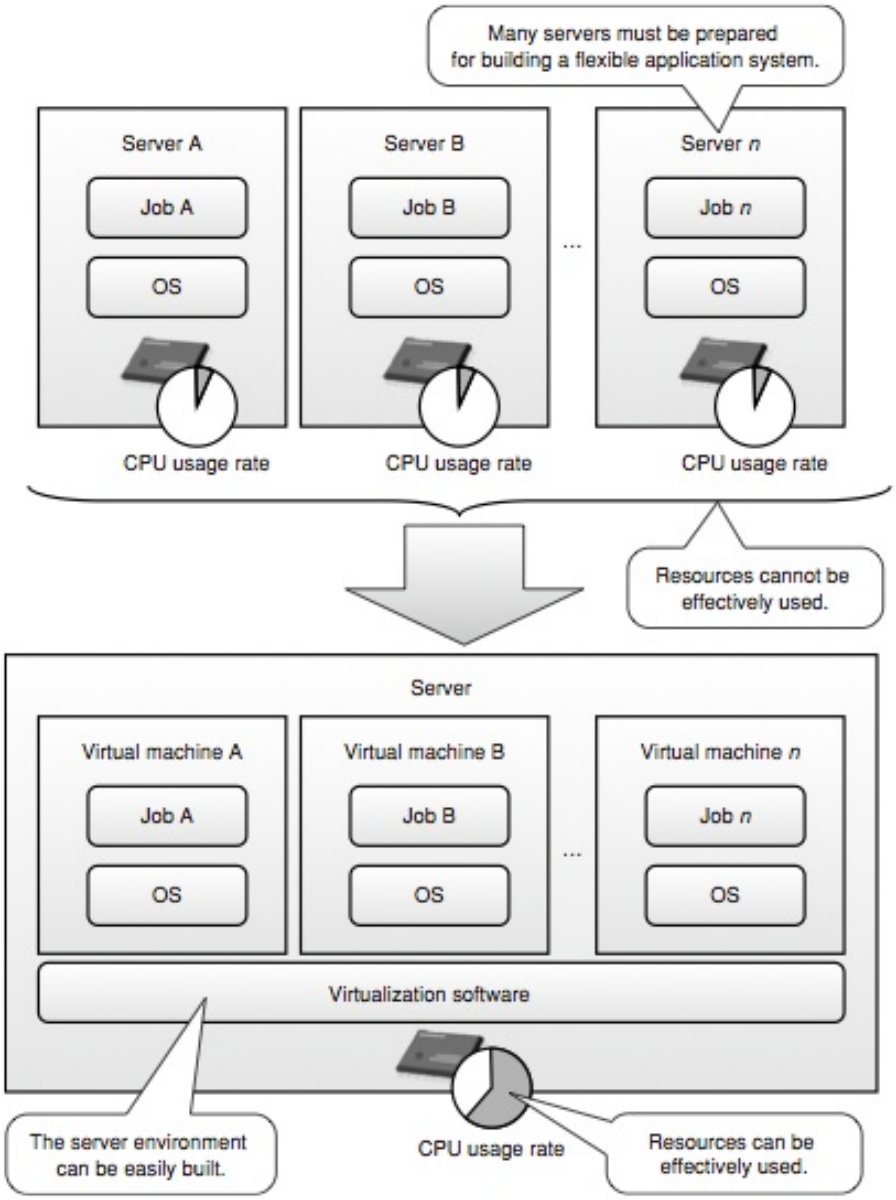

Figure 2.4: Background and effects of server virtualization [6]

2.5 VMWare Virtualization Technology

Virtualization is a technology that is increasing by every passing day in IT industry because of its number of benefits, higher utilization of expensive hardware, improved security, ease of administration, and improved data integrity. World's largest supplier of virtualization software, platforms, and tools are provided by VMWare and their products are widely used across many industries.

VMWare earlier worked on single virtual machine, but the datacenter required running

a large number of virtual machines on limited physical machine. As a top focus on resource management VMWare research on server virtualization technology. VMWare also claiming that they provide reliability to virtual machines even more than running directly on hardware. The Design of a Practical System for Fault Tolerant Virtual Machines describes one such approach to this increased reliability [2]. VMWare is now more concerned about datacenter virtualization and consistently focused on innovations. VMWare new researches are developing day by day that includes the Virtual Networking, high-speed I/O, Virtualizing Networking and Security in the Cloud, storage, Virtual Machine File System, and Scalable Virtual Machine Storage using Local Disks [2]. VMWare always focused on reducing the performance overheads.

Different kind of operating systems can be run simultaneously in virtualized architectures. In most virtualization schemes, there is a host operating system (the one machine boots with), and there are one or more guest operating systems (the virtualized machines). The virtualized operating systems permit applications written for that particular OS to run as through they were running on dedicated hardware. The VMWare products (called hypervisors) are the software that creates these virtual machines on the host OS. In some cases, the host OS is itself a VMWare product. This hypervisor is the basis for the virtualization scheme, as it permits access to the hardware from not only the host operating system but also from guest operating systems and applications being used by those guests. In most user environments, Windows would only permit one user at a time to be logged on to a machine, but in a virtualized environment many applications can have access to the CPU and memory of that machine, putting to economic use what would otherwise be sitting idle. IDC estimates that the typical Intel server is utilizing only 10-15 percent of its capacity. With virtualization, a single server could then reasonably be expected to do the works of six to ten servers, generating enormous cost savings in multi machine environments.

Although VMWare is market leader in virtualization yet still there is potential of growth. As widespread as VMWare is at the moment, the potential for growth is still there. Every new machine installation, whether it's in a Windows, OSX or Linux environment, is a potential VMWare seat. On top of that, there are many data centers that are still running dedicated hardware, and these all represent potential sale waiting to happen. The company's finances is exceptionally strong, with $3.3 billion in cash, zero short-term debt, and rock-solid net margins in the midteens [8].

Its more than a decade that VMWare is focusing on server virtualization, datacenters, and cloud computing concepts and due to that long and mature enough research of VMWare they have brought data center virtualization to more than 230,000 customers around the world [2].

2.5.1 VMWware ESXi

The ESXi is the most advanced hypervisor architecture of VMWare. VMWare Inc. provides the detail of VMWare ESXi [9]. ESXi is a Bare Metal hypervisor and directly installed on top of the physical machine. ESXi was introduced in 2007 by VMWare to deliver industry-leading performance and scalability while setting a new bar for

reliability, security and hypervisor management efficiency.

2.6 Red Hat Virtualization Technology

The kernel-based Virtual Machine (KVM) [33] is a native virtualization solution for Linux on x86 hardware supporting virtualization extensions (Intel VT or AMD-V). It has been added to the mainline Linux kernel [34]. KVM is initially developed and sponsored by Qumranet(formerly Comanet) in Israel [36]. After adding the kernel module into Linux, all the Linux standard kernel and its hardware supported virtualization is available for KVM support. Red Hat virtualization is clamming the high performance and scalability [33]. Further they claim that Red Hat Enterprise Virtualization for Servers gives near Bare Metal and even better than Bare Metal performance.

Che et al. [36] further describes the KVM architecture and added that "it implements virtualization by augmenting the traditional kernel and user mode of Linux with a new mode named guest. The guest mode has its own kernel and user mode and answers for code execution of guest operating systems. The I/O devices cannot be accessed with the guest mode and have to shift into user mode. The composition of KVM is based on two components. Kernel module that controls virtualization of hardware resources using /dev/kvm and kill command. With /dev/kvm, guest operating system may have its own address space allocated by the Linux scheduler. The physical memory mapped for each guest operating system is actually the virtual memory of its corresponding process. A set of shadow page tables is maintained to support the translation from guest physical address to host physical address. The second user space module takes charge of the virtualization of I/O by employing a lightly modified QEMU to simulate the behavior of I/O or sometimes necessarily triggering the real I/O. KVM also provides a mechanism for user-space to inject interrupts into guest operating system. Any I/O request of guest operating system is trapped into user-space and simulated by QEMU".

2.7 Background Material and Previous Work

Virtual machine monitors for the purpose of performance comparison is extensively studied by different researchers [13] [24] [37] [38] [39] [40]. VMWare performance is measured by Barham et al. [24] where they compare VMWare with Xen. KVM and Xen performance comparison was made by Deshane et al [43]. He compares the performance, isolation and scalability with CPU, kernel compile and Iozone tests.

Virtualization introduces an additional layer to access the lower layer resources by the different multiple higher level simultaneously [35]. Although different virtualization technology implementation is very effective but also yet they add significant management overhead. Virtual machine monitor is the major component of the virtualization system that caused the performance compromise for virtual machines. So its important to measure and analyze the performance of this core component [36].

KVM is the most easiest to install as compare to other virtualization technologies but VMWare is entirely different. KVM provides its own CLI interface which is not user

friendly and offer restricted advance features directly to users, such as power management or quick memory adjustment. VMWare is more user friendly as it provides the GUI along with a Web-based ActiveX client interface that allows users to easily operate the VMWare host remotely [44].

2.7.1 Virtualization Overhead

Virtualization technologies add up overhead because of additional abstraction layer between hardware resources and OS. So that to be accessed by multiple instances of operating systems (OS) to run simultaneously on a single physical host. Overhead caused performance reduction and meant the amount of processing time used by system software. According to Fuertes et al. [45] overhead is the cost difference of an application running in virtualized environment and on Bare Metal.

Some of the major virtualization overhead issues discussed by Casazza et al., [46] are listed below:

- The reported time by guest OS can be incorrect.
- Poor performance tolls can also be mislead.
- In-identical configuration can be caused incorrect information.
- Resource contention with other Virtual Machines.
- Consistency of results. Other approaches consider several factors that increase the overhead.

I/O devices, hard drive, network interface, memory can also caused overhead [47]. Besides the devices there are some other reasons that include number of virtual machines and their individual configuration that is memory, number of processor etc. Virtualization overhead also depends on the virtual machine profile as each virtual machine may have different guest operating system. All the requests for resource acquisition generated by VM wait for their turn at hypervisor [45]. Fuertes and Vergara [48] consider that particular virtualization technology may have different overhead as compared to others because of their different architecture. Different hardware resources may also caused to effect the virtualization performance. One VM overload can also be caused to have extra overhead for other VMs in the system [45]. As the virtual technology is developed to reduce the virtualization overhead as well and virtualized system performance getting closer to Bare Metal system [24] [49].

Although virtualization provides lot of benefits for its better utilization physical resources yet it also has little extra burden. In a normal system the resources of the system is only utilized by the single operating system, whereas in virtualized setup resources are competed by multiple operating systems.

The virtual machines are required a system hardware resources which cannot directly be able to approach them, but hypervisor is the one who works for virtual machines

operating system and allows to access them [15]. The difference between single operating system and virtualized multiple operating system is the process of accessing system resources. In single operating system path of access resources is from operating system kernel to the hardware resources, whereas in virtualized environment the request from operating system kernel is made to hypervisor. The hypervisor is working some sort of agent between virtual machines and hardware [12] [27].

2.7.2 CPU Overhead

This additional layer between hardware and virtual machines added up some unavoided CPU overhead. Although the virtualization technology provides the better utilization of system resources yet this overhead reduces the amount of available physical computing resources [10]. This new raised problem of virtualization system with CPU overhead leads towards many questions for its solution and further description. According to [10] followings are the questions that need to be explored.

- First, how to define the overhead and divide it from the normal utilization of CPU slice?
- The second is how about the relationship between the overhead and the count of VMs.
- The third is how the different types of workload on the VMs would affect the overhead.

2.7.3 Memory Overhead

Physical system memory is shared dynamically to the virtual machines. For optimize usage of virtual memory performance memory management unit (MMU) and translation lookside buffer (TLB) is included in CUP. Memory management unit needs to be virtualized for virtual machines. In this case guest OS don't have any direct access to the actual memory but virtual machine monitor takes charge for mapping physical to actual machine memory. Mapping of guest OS physical memory to actual machine memory by virtual machine monitor thus creates some overhead [50].

VMWare discusses [51] overhead of memory as ESX/ESXi memory virtualization adds little time overhead to memory accesses. Further they listed following two types of memory overhead:

- The additional time to access memory within a virtual machine.
- The extra space needed by the ESX/ESXi host for its own code and data structures, beyond the memory allocated to each virtual machine.

2.7.4 Disk I/O Overhead

Virtualization incurs greater stress on the physical resources of the system. Multiple virtual machines having different kind of operating systems when competing for resources, the hypervisor adds its overhead. In case of file I/O operation, it also have

virtualization overhead due to an abstraction layer between hardware and OS. The performance of the whole system is depended upon the disk performance. It is fact that generating disk I/O in one virtual machine slows I/O to the disk from other virtual systems [58]. In advance hardware where the virtualization supports is integral part of the system, the disk I/O overhead is also remained lower.

The diskeeper corporation [58] recommends the following suggestions for disk subsystem and partitioning strategies for best disk performance.

- Minimize the virtual disks fragmentation, allocation of disk size to monitor quota disk space.
- Host operating system on separate physical disk.
- Separate the host paging file on physical disk from the virtual disk.
- For each virtual system, creates separate logical partitions on host system specially for dynamically expanding virtual hard disks.
- High performance SCSI hard disk with either SAN or RAID back end.

Chapter 3

Benchmarking over view and tools

3.1 Benchmarking

A computerized test for measuring the properties of the particular technology is called benchmarking. The properties might include speed, performance, transfer rate, etc. Benchmarking is important before making decision to select an equipment. The equipment that is going to buy, must be tested before in the same environment and workload as in real working situation. Besides the working situation, it should also has to be tested in worst case situation. It might not always be possible due to non availability of replicated surrounding environment. This includes the actual data system that is working with. Because of privacy issues of data or huge amount of data replication of the systems data might not be possible. So the artificial workloads are needed for execution and monitoring the benchmark program [26].

For testing the characteristics of the technology for academic or research purpose, it is difficult to provide the real system configuration. In such cases benchmarks could serve the purpose to provide with close to real application systems for better results.

While implementing a system should have to consider the potential performance and cost of the system. Benchmarking provides the results that can help. Many factors are considered when benchmarking for comparison of different vendors products. Results from the benchmarking that are a reasonable match for the application and system size that you are considering. It is fact that it might be possible a costly system or a large database benchmark result may not hold a lot of relevance for deploying a small application server. Benchmarks are categorized as follows:

- **Performance focused** These benchmarks aim for the highest performance regardless of system cost.
- **Price or performance focused** These benchmarks aim for the lowest cost regardless of the system performance.

While setting up a new system one should consider both prices and performance reasonably. It might happen that good performance system can be very expensive or low budget system can be bad performance. Following can also be considered while benchmarking:

- **System architecture** 32-bit or 64-bit. x86, Itanium, POWER, etc
- **System size** The number of CPUs in the system under test
- **System configuration** Multiple clustered systems, or a single non-clustered system. Benchmarks for multi-tier applications may use different architectures and operating systems for different tiers.
- **Database size** Range from 100GB to many terabytes
- **Services** Some benchmark includes the cost of 24x7 support, others factors do not support costs into the final result

3.2 I/O benchmark tools

In the following section we will discuss different I/O benchmarking tools in brief:

3.2.1 Flexible I/O

fio is a tool that will spawn a number of threads or processes doing a particular type of I/O action as specified by the user. The typical use of fio is to write a job file matching the I/O load one wants to simulate.

3.2.2 Bonnie++

Bonnie++ is a benchmark tool for testing hard disks and file system performances. Bonnie [28] is a micro benchmark, written by Bray in 1988, 1989 and 1996, which measures sequential input and output, creating and reading a file character for character, and in 8 KB chunks and random seeks, in terms of number of bytes per second, on a file [26].

The file size is the only that is configurable, whereas all the other values are hardcoded in Bonnie. From 2000, Coker rewrite the Bonnie into Bonnie++ [29]. Bonnie also added new features for creating and deleting files, direct I/O and benchmarking ZCAV effects. Bonnie++ is made able to use several files, for datasets larger than 2 GB [26].

3.2.3 Postmark

Postmark [30] is a macrobenchmark which emulates e-mail, netnews and webbase commerce software; workloads which consists of a large number of shortlived files. A pool of a large number of text files are created when the Postmark runs for performance measurement. Whereas different Postmark operations include create, delete, read and write with random file selection. At the end the delete performance is measured while deleting the remaining files.

Postmark configuration is quite easy and simple to run and has its own pseudo-random number generator. This has an advantage when comparing results from different operating systems as it provides identical conditions across platforms [26]. Whereas the

outdated default configuration file of Postmark does not fit for current hardware and does not scale the workload. The result comparison is difficult because lack of standard and scaling workload. Postmark is not updated and maintained so the disadvantages are not likely to be corrected [31].

3.2.4 Iozone

Iozone is a file system benchmark tool. The benchmark generates and measures a variety of file operations. Iozone has been ported to many machines and runs under many operating systems. Iozone [32] is analyzed file systems, written by Norcott, with enhancements by Capps. Iozone is quite a lot of different operations for testing and can be used to many operating systems, including Sun Solaris and DragonFly. Iozone operations are listed in detail by Oppegaard in table 3.1 [26]

Table 3.1: Iozone operations for benchmarking file system

Operation	Description
Write	Create and write to a file. When creating and writing to a file, creating meta-data for mapping the location of the file and its associated data blocks on the storage medium, adds an overhead which degrades performance. Because of this overhead, writing to a pre-existing file is normally faster than creating it additionally.
Re-write	Write to an existing file. Re-writing a file is ordinarily better, performance wise, than when it has to be created first, because its meta-data already exists, and thus does not have to be created again.
Read	Read a file sequentially.
Re-read	Read the file again. Reading a file is typically faster when it has recently been read, because the data is stored in caches maintained by the operating system, file system and storage medium itself.
Record re-write	Write and re-write a section of a file. The characteristics of this test depend upon the size of the section being tested. If it is small and fit the CPU data cache, the test will show high performance. Following are a few examples of caches which the results depend upon, ordered with higher performance first: Fit in the CPU data cache and the TLB, does not fit in neither the CPU data cache nor the TLB but fit in the operating system's cache, larger than the operating system's cache.
Random read	Read a file from random locations. The size of the operating system's cache, the number of disks and seek latencies are examples of factors which can impact the results from this test.
Random write	Write to random locations in a file. The same factors affecting the results of the random read test, apply here as well.

Table 3.1: (continued)

Operation	Description
Backward read	Read a file backwards. Many operating systems have optimizations for reading files forwards, but few have implemented enhancements to backward reading.
Stride read	Read a file at a stride offset. The stride read test read chunks the <stride parameter offset >apart, i.e., read x KB, seek <stride parameter >, read x KB, read <stride parameter >and so on. This test can be used to tell how well a RAID setup perform when the stride is aligned and unaligned with the RAID's stripe boundary.
fwrite(3)	Write a file using fwrite(3). fwrite(3) is a standard C function which performs buffered write operations. This function can reduce the number of system calls, and increase transfer size. Like regular write, fwrite(3) creates a new file and has the overhead of creating meta-data.
Re-fwrite(3)	Write to an existing file with fwrite(3). Like re-write, but uses the buffered fwrite(3) function.
fread(3)	Read a file using fread(3). This test is similar to the read test, but the buffered fread(3) function is used.
Re-fread(3)	Read a recently read file using fread(3). Using fread(3), the file which was recently read by the previous test is read again. Performance should be higher than the first read, as the file is now stored in caches.

Further can also help to determine the platform optimization for certain workloads, or its more generic to provide the information for purchasing an equipment. Oppegaard [26] quoted that computers are acquired for particular task but as the time past the applications and task are changed which results an optimized system is not guaranteed to perform adequately during its life span.

3.3 Memory benchmark tools

3.3.1 bandwidth

One way to analyze the memory is bandwidth. Bandwidth can test the memory in different ways that includes sequential read and write for CPU level 1 and 2 cashes besides the main memory and video memory. Bandwidth test the performance and speed of memst, memcopy and bzero routines. Bandwidth is handy to figure out the performance related problems of memory [52].

3.3.2 mbw (Memory Band Width)

MBW determines the "copy" memory bandwidth available to user space programs. Mbw is another memory benchmarking tool that uses two arrays in memory. Theses arrays are twice the size of physical memory that is under test. User can give the size

of the array through command line to create arrays. Then different methods includes memory copy function memcpy(), dumb copy by element coping, and by block copy of a defined block size can be used [53].

3.3.3 RAMSpeed

RAMSMP is tool written in C and used to benchmark cache and memory to determine the bandwidth of a system's various memory components. RAMSMP can run 18 memory intensive tests at a time and each of them measure a different aspect of the systems memory performance. These tests are focused on testing the reading, writing and data manipulation bandwidth of memory operations on the system. Floating point and integer operations are used by this benchmark as most of the scientific applications used these data types [54].

Ramspeed and RamSMP work on testing the CPU processing for integers, floating point numbers and on Intel based systems the extensions of MMX and SSE instructions. For each test group the tool executes three sub-tests: reading of the cache, writing of the cache and the last tries to give a simulation of CPU and cache interaction by testing the following copy, scale, add, and triad [55].

RAMSMP first FLOATmem and INTmem test first copy either floating point or integer data from one to another memory location to test memory bandwidth with respect to calculations using large blocks of data. Different numerical operations on both sets of data including scaling, adding and a combination of these two operations, and store the results in the memory. At the end of all these operations, obtained bandwidth shows the rate at which the memory is able to manipulate different blocks of numerical data.

While testing the reading and writing bandwidth of the systems memory, FLOATmark and INTmark test read and write blocks of exponential increasing data sizes to the memory. Block size starts from 2 KB to the maximum size of block which is by default 32 MB but can be changed up to 2 GB. The bandwidth at which each block of data was written or read from memory allows distinction between each cache level and RAM presents on the system being benchmarked, as well as the bandwidth at which integer and floating-point data is typically written to memory. Usually L1 cache is observed with fastest bandwidth whereas the RAM reflects the slowest bandwidth while reading and writing the data.

Care must also be taken while specifying the size of the data that is either manipulated at different memory levels. The block of data that exceeds the size of the physical memory of the machine can be caused overload its resources. The size of the block should not be exceeded the memory and at the same time should be larger enough to write the data to ram.

3.4 CPU benchmark tools

3.4.1 CPU Free BenchMark (former CPUMark)

CPU benchmark is required to know the performance of the CPU specially in virtualized environment when more then one OS routines are competing for CPU. CUP Free BenchMark is open source performance measurement tool that tests the registries, ALU (Arithmetic Logic Unit) and FPU (Floating Point Unit). All the tests are based on some equations and operations depending upon the test.

CPU Free BenchMark generate the reliable test results by conducting the 3 identical tests for each CPU part. The data from each identical test is generated in seconds and milliseconds and means of all the three tested is produced. The average of three tests reduce the chance of wrong data collection. The final test results are composed on the following equation.

The final score = 40% Score Test1 + 80% Score Test2 + 80% Score Test3

The CPU Free BenchMark application is opened for all kind of processors type to test their performance and not a specific type [56].

3.4.2 SysBench

SysBench is an open source tool, that can be used for almost all kind of operating system and multi threaded benchmark. SysBench is made to evaluate OS parameters that are important for a system running a database under intensive load [60]. According to Kopytov SysBench is quite fast and gives system performance without configuration of database or even without database installation on the system.

SysBench is simple tool to use and it runs specified number of threads by default that is one. Different test options are available that can be mentioned with test name. SysBench runs until the defined prime number reached that is by default 20000 but can be changed according to specific requirement.

3.4.3 UnixBench

UnixBench was first started in 1983 at Monash University is an open source tool that provides performance of Unix systems in various aspects of system performance specifically for CPU. UnixBench produces the results as an index values after comparing the test results with baseline system. George SPARCstation 20-61 with 128 MB RAM, a SPARC storage array, with Solaris 2.3 set rating as 10 is used as baseline system since 1995. The system that score 600 is considered as 60 time faster than this baseline system. The reason for producing the indexed results, because it makes easy to discuss for analysis point of view. It also produces the overall index after combining the entire set of index values. The first run of UnixBench generates the results for first CPU and then it runs N copies for N number of CPUs. UnixBench [57] is enabled to assess the following in table 3.2:

- the performance of the system when running a single task
- the performance of the system when running multiple tasks
- the gain from the systems implementation of parallel processing

The detail of UnixBench different tests are taken from project home page [59] and given in the table 3.2:

Table 3.2: UnixBench for CPU benchmarking

Test	Description
Dhrystone	Developed by Reinhold Weicker in 1984 and used to measure and compare the performance of computers. This test focuses with string operations but no floating point operation. It is heavily influenced by hardware and software design, compiler and linker options, code optimization, cache memory, wait states, and integer data types.
Whetstone	Performance in terms of speed and efficiency of integer and floating point operations is measured in this test. This test based on different trigonometric functions including sin, cos, sqrt, exp, and log are used as well as integer and floating-point math operations, array accesses, conditional branches, and procedure calls.
Execl Throughput	Number of execl calls that can be performed per second is measured using this test. Execl is part of the exec family of functions that replaces the current process image with a new process image.
File Copy	File copy measures the rate at which data can be transferred from one file to another using different buffer size. The file read, write and copy tests captures the number of characters that can be written, read and copied in a specified time.
Pipe Throughput	Communication between processes can be done while using pipe that is simplest form. Pipe throughput is the number of times (per second) a process can write 512 bytes to a pipe and read them back.
Pipe-based Context Switching	The exchange of integers through pipe between two process is measured using this test. The pipe-based context switching test is more like a real-world application.
Process Creation	This test measures the number of times a process can fork and reap a child that immediately exits. It is actually creating process control blocks and memory allocations for new processes and directly applies to memory bandwidth. This benchmark is used to compare various implementations of operating system process creation calls.
Shell Scripts	The shells scripts test measures the number of times per minute a process can start and reap a set of one, two, four and eight concurrent copies of a shell scripts where the shell script applies a series of transformation to a data file.

Table 3.2: (continued)

Test	Description
System Call Overhead	This estimates the cost of entering and leaving the operating system kernel, i.e. the overhead for performing a system call. It consists of a simple program repeatedly calling the getpid (which returns the process id of the calling process) system call. The time to execute such calls is used to estimate the cost of entering and exiting the kernel.

Chapter 4

Approach

The first section of this chapter describes the overview of the hardware under test and the OS of virtual machines used. The detail of operation of the test and how the data collected using different benchmarking tools discussed in the end of this chapter.

4.1 System Hardware Specification

Dell Poweredge R710 server is used for benchmarking purpose is one of the requirement of this project. This server have following specifications:

CPU
Two 2.13 GHz Quad-Core Processors
Intel®Xeon®Processor L5630
Number of Cores: 4
Number of threads: 8
Clock speed: 2.13 GHz
Buss speed: 5.86 GT
L1: 256 KB, L2: 1 MB, L3: 12 MB
Instruction set: 64-bit

Memory
Size: 72 GB with speed of 1067 MHz

Hard disk
Two SCSi disk with 140 GB each.

Software
Red Hat Enterprise Linux 6.1
Kernel version: 2.6.32-220.7.1.el6.x86_64
KVM (Kernel based virtual machine)
VMWare ESXi 5.0

4.2 Method for conducting the tests

Different benchmarks are used to test the different aspects of the system under test that is Dell Poweredge R710. As the purpose of this research work is to test the Red Hat Enterprise Linux 6.1 with respect of Disk I/O, memory, and CPU performance in following three cases:

- Bare Metal
- KVM virtual machine
- VMWare virtual machine

The system under test have two SCSi hard disk of 140 GB, one for Bare Metal and KVM and second for VMWare. Being a part of RHEL Kernel, KVM do not need separate hard disk. First of all the RHEL 6.1 is installed on the server with standard installation [41]. In next step install the Iozone 3.3 [32] for disk I/O, Ramspeed 2.13.1 [55] for memory and UnixBench 5.1.2 [59] for CPU benchmarking. Some additional repositories are needed to install these packages. After collecting the data from Bare Metal by using the benchmarking tolls, KVM is activated on RHEL 6.1 [42]. After KVM installation, a virtual machine created and RHEL 6.1 OS installed. After installation of benchmarking tool data is collected from KVM virtual machine.

VMware vSphere Hypervisor (ESXi) 5.0 is installed on second hard disk. To access the VMWare, VShpere client installed and virtual machine is created. RHEL 6.1 along with benchmarking tool installed on VMWare virtual machine for data collection purpose. The the configuration of KVM virtual machine and VMWare Virtual machine was similar. The version of RHEL OS, Iozone, Ram Speed and UnixBench was same on both the virtual machines. To make the test result more reliable and adding authenticity in the data collected, the test was repeated several times. Iozone is repeated 70 time, Ramspeed is repeated 25 times, and UnixBench was repeated three times. Average of all the repetition was used for analysis purpose.

Process of conducting the performance test is long and time taking job, to make it easier an automatic procedure is adopted. Perl script is used for automation process, that helped to repeat the test for number of desired time and store the data in separate text file. As in all three cases the operating system will be the same, so once the script is written for benchmarking tool, it can be used for all three types of cases that is Bare Metal, KVM virtual machine and VMWare virtual machine. For Iozone and Ramspeed script were written as it repeats for number of times. But in case of UnixBench it repeats for three time that does not required a script.

Test environment for benchmarking is described in next section.

4.2.1 Implementation of Iozone test

Iozone will be used to test the I/O performance on Red Hat Enterprise Linux 6.1. Iozone is known tool for disk I/O operations benchmarking and provides the various useful results to determine the I/O of the disk. For better results plan is to test the

different file size and will use 1 MB, 64 MB, 128 MB, 256 MB, 512 MB and 1 GB. Each file was created using a record size that is the amount of data written into a file during a single IO operation is 4 KB. For each size the standard test was repeated every 5 minutes several times by using a shell script. For more reliable results, test is repeated 70 time and data is collected. Then average of these 70 repeated test was used for graphical presentation and analysis purpose. The test results were directed to a file which later on was parsed by another shell script to write the data to a new file on the form that can be imported in R. Such files are easy to use when creating graphs with R.

Iozone have different options to use for changing the file size, record set, type of test etc. Short description of available options are:

- -s option is the size of the file to be measured
- -r is the record size, in Kbytes
- -i is ued to specify the type of performance but in this particular case we did not used -i that mean to measure all kind of performance
- -R option generates an Excel report
- -c includes close() in time calculation

Iozone tests the several types of properties for the I/O operation that includes: write, rewrite, read, reread, random read, random write, backward read, record rewrite, stride read, fwrite, frewrite, fread, freread. For this research work I/O performance is tested against all the properties of the Iozone performance test. Iozone produces the test results calculated in kilo bytes but for better readability the scale the data changed into mega bytes. Single run of Iozone was:

iozone example

```
1 iozone -s 1024M -r 4k -Rac >> iozonetest.txt
```

The result of the above command is stored in text file and shown under:

The result of Iozone command

```
1       Iozone: Performance Test of File I/O
2            Version $Revision: 3.394 $
3            Compiled for 64 bit mode.
4            Build: linux
5
6       File size set to 1048576 KB
7       Record Size 4 KB
8       Excel chart generation enabled
9       Auto Mode
10      Include close in write timing
11      Command line used: iozone -s 1024M -r 4k -Rac
12      Output is in Kbytes/sec
13      Time Resolution = 0.000001 seconds.
14      Processor cache size set to 1024 Kbytes.
15      Processor cache line size set to 32 bytes.
16      File stride size set to 17 * record size.
17
18   KB      reclen   write    rewrite     read      reread    random read   random write   bkwd read
19   1048576 4        538044   835397  3616565   3622749   2348734         748127         3230143
```

```
20
21  record rewrite   stride read   fwrite    frewrite    fread       freread
22    1068210         2986758     828436   830210   3509842    3517329
23
24  iozone test complete.
25  Excel output is below:
26
27  "Writer report"
28          "4"
29  "1048576"   538044
30
31  "Re-writer report"
32          "4"
33  "1048576"   835397
34
35  "Reader report"
36          "4"
37  "1048576"   3616565
38
39  "Re-Reader report"
40          "4"
41  "1048576"   3622749
42
43  "Random read report"
44          "4"
45  "1048576"   2348734
46
47  "Random write report"
48          "4"
49  "1048576"   748127
50
51  "Backward read report"
52          "4"
53  "1048576"   3230143
54
55  "Record rewrite report"
56          "4"
57  "1048576"   1068210
58
59  "Stride read report"
60          "4"
61  "1048576"   2986758
62
63  "Fwrite report"
64          "4"
65  "1048576"   828436
66
67  "Re-Fwrite report"
68          "4"
69  "1048576"   830210
70
71  "Fread report"
72          "4"
73  "1048576"   3509842
74
75  "Re-Fread report"
76          "4"
77  "1048576"   3517329
```

This single run gives the detail output of the Iozone test. In the beginning of the file it shows the command used, file size, record size, tike taken, and number of kilo bytes per second. The second half of the result shows the performance of each 13 operations that Iozone perform during disk I/O operation. The amount of data per second is shown

in kilo bytes is quite larger and converted into mega bytes for better readability. Two different files for Iozone test is used to collect the data. Ionozetest.sh will run the Iozone test with 70 time repetition and transfer the results in iozonetest.txt file using the following command.

iozonetest.sh

```
./iozonetest.sh
```

The iozonetest.txt file contains the raw data of Iozone test with other unnecessary information as well. For the purpose of generating the graph and data summary that includes mean, median, maximum and minimum values, first and third quartiles in R, iozonetest.txt file passed to fileseprator.sh using following command

fileseprator.sh

```
./fileseprator.sh iozonetest.txt
```

4.2.2 Implementation of Ram speed test

RAMSMP provides different options to test the different memory level in the system. In this case as the system under test have a larger physical memory, so large size of the block is used that is 2 GB which is maximum available size in ram speed. The maximum block size can be determined by using the -m option in ram speed. Option -b is used to determines the type of test in particular test that can be variated with integer. In this particular case four options are used to test the memory performance that includes integer read and write and float read and write, hence used from 1 to 4 with -b option. Following is the ramsmp command that is used to test the memory with -b1 that is integer writing with block size of maximum 2 GB.

Ramspeed example

```
[root@naveed ramsmp-3.5.0]# ./ramsmp -b1 -m 2048
```

The above command will benchmark the memory with Integer write operation by using the block size starting from 2 KB to 2 GB with exponentially increment. After each block size it calculate the bandwidth of memory of per second mega bytes that shown in the last column.

The result of Ramspeed command

```
8GB per pass mode, 2 processes

INTEGER & WRITING       1 Kb block: 37841.87 MB/s
INTEGER & WRITING       2 Kb block: 38115.23 MB/s
INTEGER & WRITING       4 Kb block: 38123.18 MB/s
INTEGER & WRITING       8 Kb block: 38060.18 MB/s
INTEGER & WRITING      16 Kb block: 38113.54 MB/s
INTEGER & WRITING      32 Kb block: 37852.34 MB/s
INTEGER & WRITING      64 Kb block: 33881.47 MB/s
INTEGER & WRITING     128 Kb block: 33804.33 MB/s
INTEGER & WRITING     256 Kb block: 32621.90 MB/s
INTEGER & WRITING     512 Kb block: 26705.57 MB/s
INTEGER & WRITING    1024 Kb block: 26019.54 MB/s
INTEGER & WRITING    2048 Kb block: 25993.22 MB/s
INTEGER & WRITING    4096 Kb block: 26005.46 MB/s
INTEGER & WRITING    8192 Kb block: 25998.06 MB/s
```

```
INTEGER & WRITING     16384 Kb block: 12981.12 MB/s
INTEGER & WRITING     32768 Kb block: 11676.65 MB/s
INTEGER & WRITING     65536 Kb block: 11558.14 MB/s
INTEGER & WRITING    131072 Kb block: 11498.66 MB/s
INTEGER & WRITING    262144 Kb block: 11372.35 MB/s
INTEGER & WRITING    524288 Kb block: 11195.43 MB/s
INTEGER & WRITING   1048576 Kb block: 10753.74 MB/s
INTEGER & WRITING   2097152 Kb block: 9966.57 MB/s
```

The first column in the above result is INTEGER & WRITING because of the -b1 option. In second column, block size is mentions with exponential increment and maximum of 2 GB because of -m 2048 option used in the ram speed command. Whereas in last column, memory bandwidth in MB per second is generated after test.

As we observed while the block size increased the speed of writing decreased. For block size 1 KB to 32 KB, the average writing speed is above 38000 MB per seconds. That is probably L1 cache memory type which is fastest in the system. From 64 KB to 256 KB, average bandwidth is above 33000 MB and could be L2 cache of the system that is second highest speed after L1. Further from block size 512 KB to 8 MB with average bandwidth is above 26000 MB per second and could be L3. From 16 MB to 2 GB block size the average bandwidth is above 11000 MB per second and that could be the ram with slowest rate. The test is repeated 25 time and average results is used to make sure about the data perfection in each of the four cases that is Integer and Float reading and writing.

To the process automated, a perl script was written in three files. The first file ramspeed.sh run the ramsmp command for 25 time with different options of integer and float with writing and readings and store the data in text file. The ramspeed.sh is run using following command,

ramspeed.sh

```
[root@naveed ramsmp-3.5.0]# ./ramspeed.sh
```

After finishing the first test with 25 time iteration, the text file containing the data is passed to an other scripting file called datagenerator.sh. This script will separate the text file according to their different nature and block. An other script called ramsinglefile.sh then take this separated file and convert the data in format that will be later used in R for graphical presentation and analysis. Following command will be used for this purpose,

ramsinglefile.sh

```
[root@naveed ramsmp-3.5.0]# ./ramsinglefile.sh ramdata.txt
```

4.2.3 Implementation of UnixBench test

UnixBench is quite straight forward to implement this test is not repeated that much time as compared with Iozone and ram speed. Three run of UnixBench is done using the following command,

UnixBench command

```
[root@naveed unixbench]# ./Run
```

UnixBench just need to run from its folder by running the above command. It will generate the results itself by doing nothing more. The results are shown as under,

UnixBench result

```
 BYTE UNIX Benchmarks (Version 5.1.3)
   System: KVMVM: GNU/Linux
   OS: GNU/Linux -- 2.6.32-131.0.15.el6.x86_64 -- #1 SMP Tue May 10 15:42:40 EDT 2011
   Machine: x86_64 (x86_64)
   Language: en_US.utf8 (charmap="UTF-8", collate="UTF-8")
 Benchmark Run: Fri Apr 13 2012 16:49:19 - 17:17:39
 16 CPUs in system; running 16 parallel copies of tests

 Dhrystone 2 using register variables         174454949.1 lps   (10.0 s, 7 samples)
 Double-Precision Whetstone                       34426.3 MWIPS (9.7 s, 7 samples)
 Execl Throughput                                 15058.4 lps   (29.8 s, 2 samples)
 File Copy 1024 bufsize 2000 maxblocks           324520.3 KBps  (30.0 s, 2 samples)
 File Copy 256 bufsize 500 maxblocks              86584.3 KBps  (30.0 s, 2 samples)
 File Copy 4096 bufsize 8000 maxblocks           978506.9 KBps  (30.0 s, 2 samples)
 Pipe Throughput                                 6397344.2 lps   (10.0 s, 7 samples)
 Pipe-based Context Switching                    1269602.1 lps   (10.0 s, 7 samples)
 Process Creation                                 47100.3 lps    (30.0 s, 2 samples)
 Shell Scripts (1 concurrent)                     27155.0 lpm    (60.0 s, 2 samples)
 Shell Scripts (8 concurrent)                     3750.5 lpm     (60.1 s, 2 samples)
 System Call Overhead                            6921005.5 lps   (10.0 s, 7 samples)

 System Benchmarks Index Values               BASELINE       RESULT        INDEX
 Dhrystone 2 using register variables         116700.0    174454949.1      14949.0
 Double-Precision Whetstone                       55.0        34426.3       6259.3
 Execl Throughput                                 43.0        15058.4       3502.0
 File Copy 1024 bufsize 2000 maxblocks          3960.0       324520.3        819.5
 File Copy 256 bufsize 500 maxblocks            1655.0        86584.3        523.2
 File Copy 4096 bufsize 8000 maxblocks          5800.0       978506.9       1687.1
 Pipe Throughput                               12440.0      6397344.2       5142.6
 Pipe-based Context Switching                   4000.0      1269602.1       3174.0
 Process Creation                                126.0        47100.3       3738.1
 Shell Scripts (1 concurrent)                     42.4        27155.0       6404.5
 Shell Scripts (8 concurrent)                      6.0         3750.5       6250.8
 System Call Overhead                          15000.0      6921005.5       4614.0
                                                                          ========
 System Benchmarks Index Score                                             3432.1
```

The developer of the UnixBench claims that analyzing the test results are depend on the requirement of user that what they want to measure. UnixBench focus testing the compiler performance and can be handy to know the CPU speed measurement and compiler quality. As the system speed is major concern which depends on compiler, then UnixBench results gives the answer to that question.

All the above three benchmarking tools are implemented in similar way to Bare Metal, KVM virtual machine and VMWare virtual machine. After collecting the data by using the benchmark tools with the help of scripting, analysis will be made using different graphs.

Box plot is the best option to present that kind of data. Box plot is convenient way of presenting the groups of numerical data. Data summary that includes minimum

sample, maximum sample, lower (Q1) and upper (Q3) quartile and median of the data. Outliers are also indicated in graph. Different part of the box spacings indicate the spread and skewness in the data. A statistical program R is used to make a box plot. R is a open source software for statistical computing and graphics. several options are available to present the data using R. The data manipulation ins R is simple and faster and used many research work analysis [61]. The code written in perl script was designed that the generated text file can be used in R for obtaining the summary of data and box plot for analysis and discussion purpose.

Chapter 5

Results and Discussion

This chapter presents the graphical review of the data with the analysis and discussion. The first section of this chapter shows the results by using the graphs for each benchmarking tool. In last section, the analysis of the data was made by using various statistical methods. The data from Iozone and Ram Speed was collected after a large number of runs because of variation in data. In case of UnixBench, the data was collection after 3 runs because of small variation in data was observed. The data was collected with the help of the perl scripts.

The gathered data was imported to R for analysis purpose. For presentation of data, box plot was used. Box plot gave the more accurate information about the data. The skewness of the data can easily be observed in the box plot. The reliability of the data was achieved by several repetitions of the each benchmarking tool. For Iozone 70, for Ram Speed 25 and for UnixBench 3 times data was collected. Box plot for all the three systems of each benchmark presented in **Appendix A, B and C**. For analysis purpose, few interesting cases were selected and discussed by using the different statistical measurements.

5.1 Iozone

This section contains the average of the data obtained after 70 runs of Iozone. The average value was used to generate the combined bar graphs of Bare Metal, KVM and VMWare. The detail of the data for Iozone is presented in box plots as **Appendix A**.

Iozone was used to benchmark the I/O performance of the different guests. Different file sizes of 1 MB, 64 MB, 128 MB, 256 MB, 512 MB and 1 GB were used to calculate the performance with different workload. Following graphs shows the performance of Bare Metal, KVM VM, and VMWare VM.

5.1.1 Write

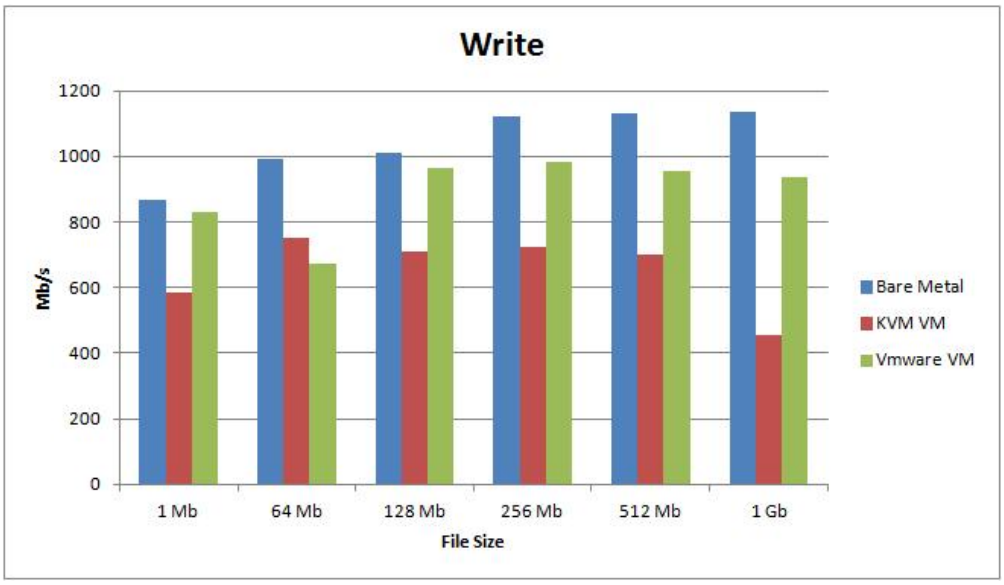

Figure 5.1: Iozone average write

5.1.2 Re-Write

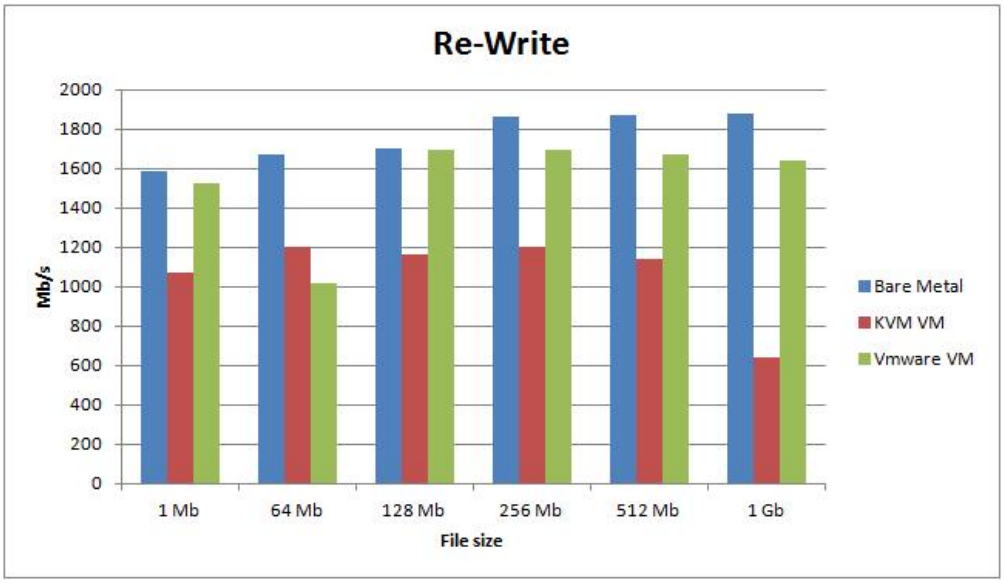

Figure 5.2: Iozone average re-write

The performance of Iozone with writing and re-writing was illustrated in graph 5.1 and 5.2. Performance of Bare Metal was remained on top while comparing with VMWare and KVM. Whereas VMWare performance was observed closed to Bare Metal. It observed that both the graphs show increased in performance while the file size increased

from 1 MB to 1 GB. As stated in Iozone properties 3.1, re-writing is usually faster than writing. It was evident from graphs that re-writing was almost twice faster than writing. The file size of 64 MB, KVM shows 11% better performance than VMWare.

5.1.3 Read

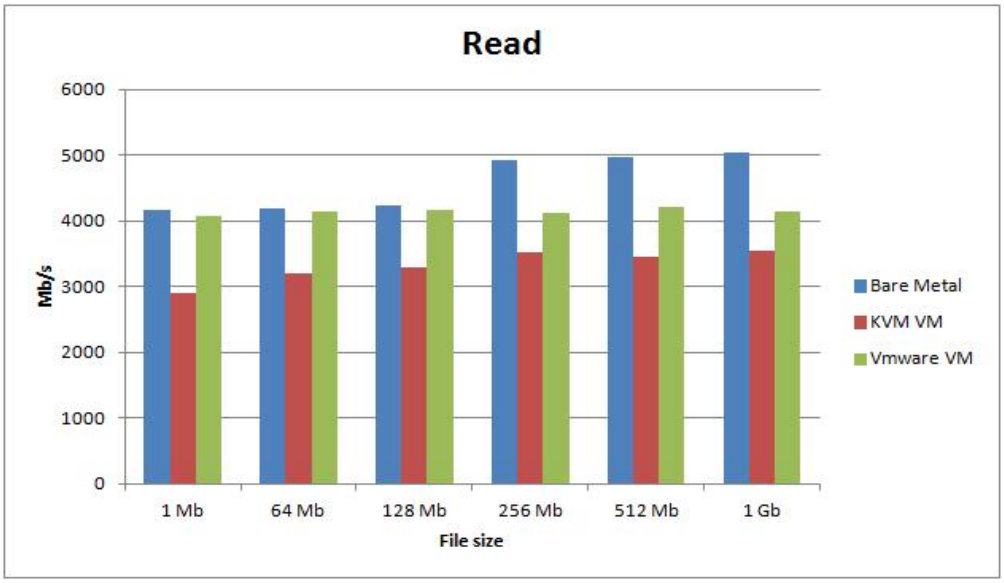

Figure 5.3: Iozone average read

5.1.4 Re-Read

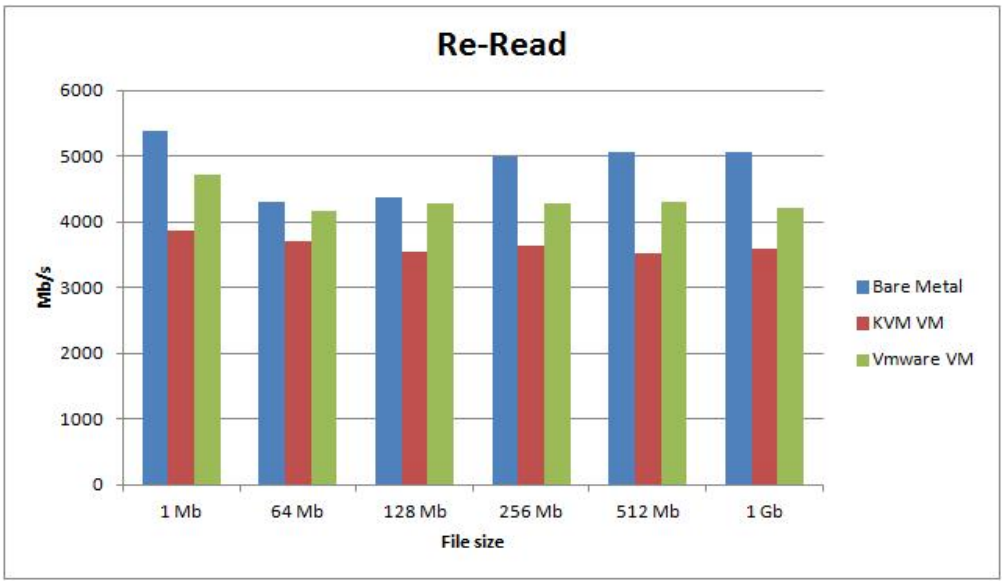

Figure 5.4: Iozone average re-read

As the properties of Iozone 3.1 said that re-read is quite faster than read, because the file is stored in cache of the system. By observing both the graphs 5.3 and 5.4, it was clear that performance of re-read was better than read. Bare Metal show increased performance in case of read as compared from file size 1 MB to 1 GB. Whereas in case of re-read it shows increment from 64 MB to 1 GB file size. KVM show roughly constant increment in case of read and decrement in case of re-read when increasing the file size from 1 MB to 1 GB. The performance of VMWare in case of read was remained constant with increase of file size. VMWare in smaller file sizes in read gave outstanding performance which was very close to Bare Metal system.

5.1.5 Random Read

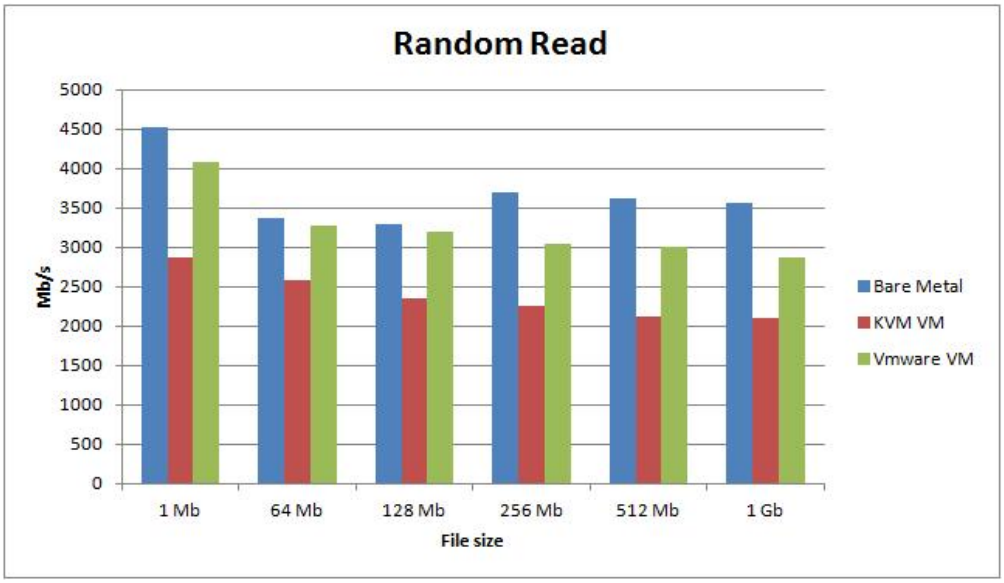

Figure 5.5: Iozone average random read

Random read as illustrated from graph 5.5, KVM and VMWare had shown continuous decrease in performance while increasing the size of the file. VMWare performed better than KVM with the difference of 27%, 36%, 35%, 41% and 36% in case of 64 MB, 128 MB, 256 MB, 512 MB and 1 GB file sizes respectively.

5.1.6 Random Write

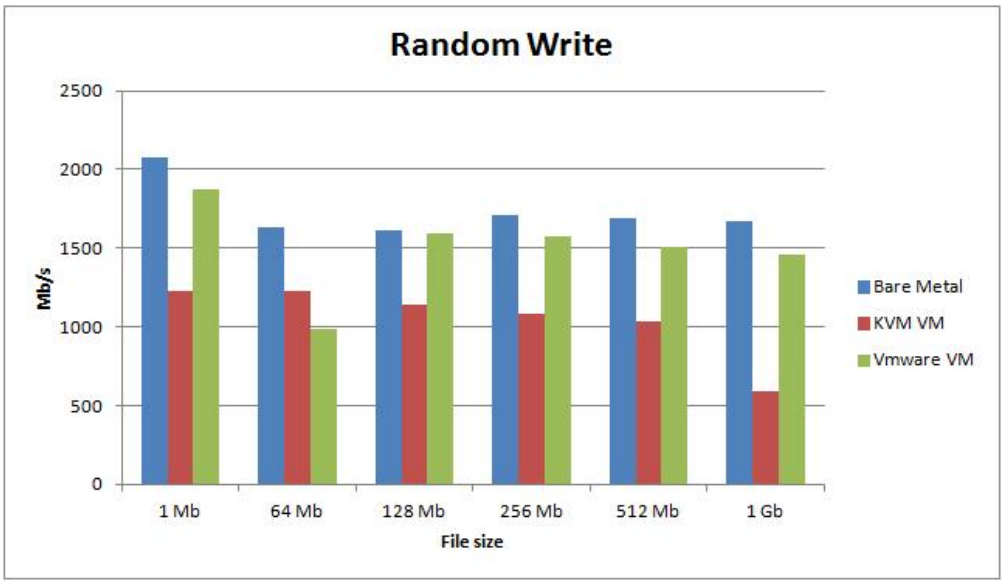

Figure 5.6: Iozone average random write

In graphs 5.6 VMWare out performed KVM in all the file sizes with average of 45%. In case of 64 MB file size KVM perform better than Vware.

5.1.7 Backward Read

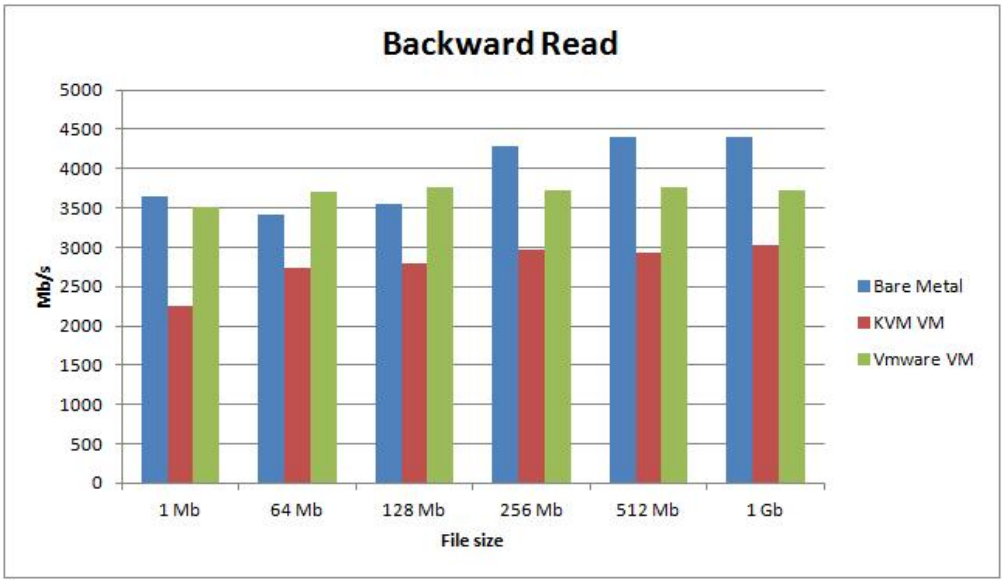

Figure 5.7: Iozone average backward read

In Backward read 5.7 Bare Metal, KVM and VMWare has shown continuous increase in performance. In case of 64 MB and 128 MB, VMWare was on top with 3707 and 3767 as compared with Bare Metal 3409 and 3543. That was quite unusual results due to some unknown reason.

5.1.8 Record Rewrite

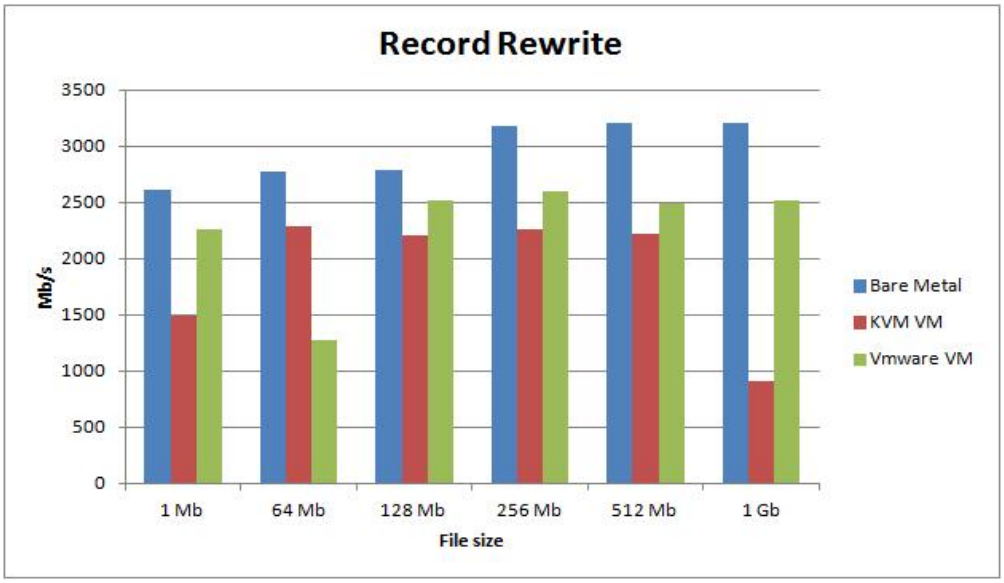

Figure 5.8: Iozone average record re-write

5.1.9 Stride Read

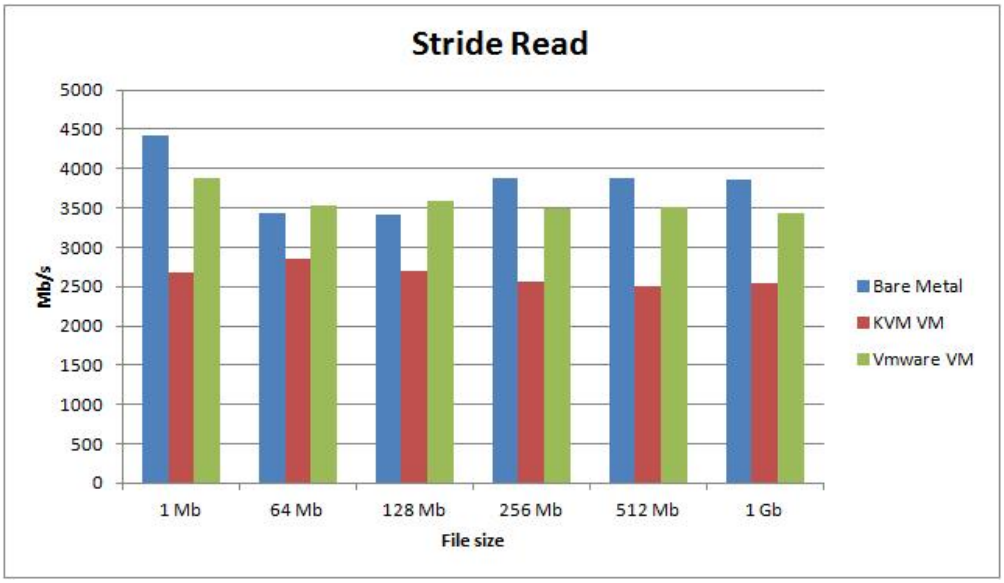

Figure 5.9: Iozone average stride read

5.1.10 Forward Write

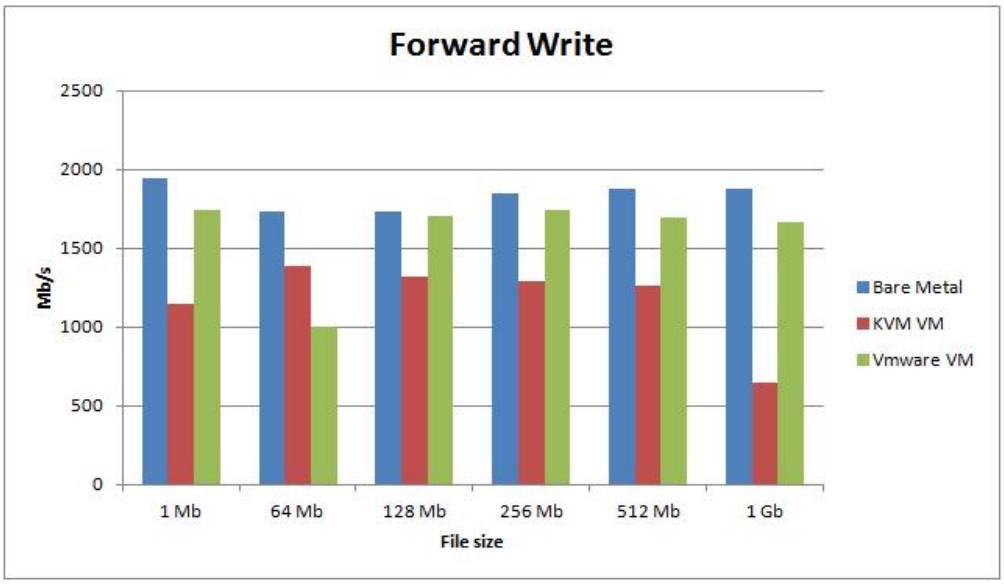

Figure 5.10: Iozone average forward write

5.1.11 Re-Forward Write

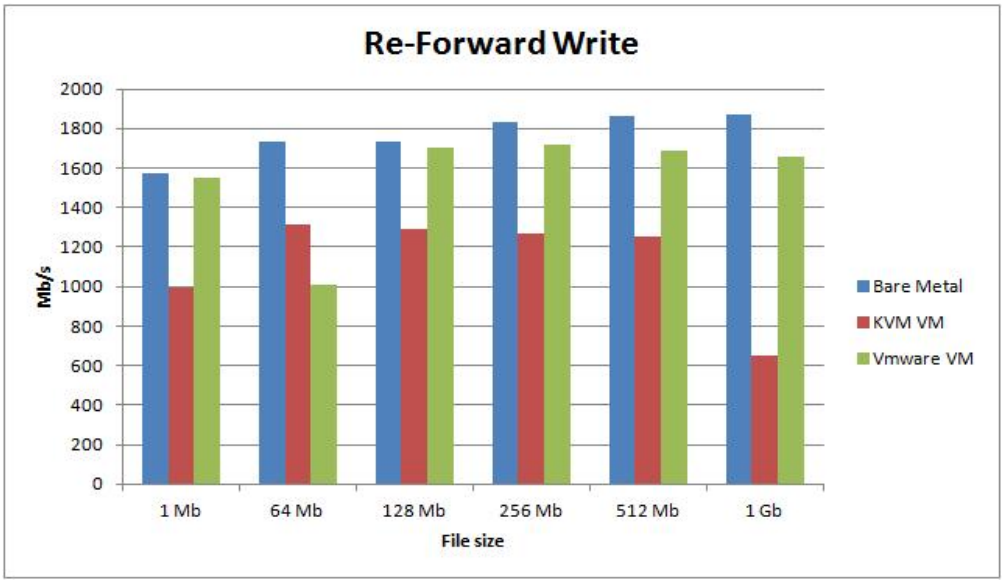

Figure 5.11: Iozone average re-forward write

5.1.12 Forward Read

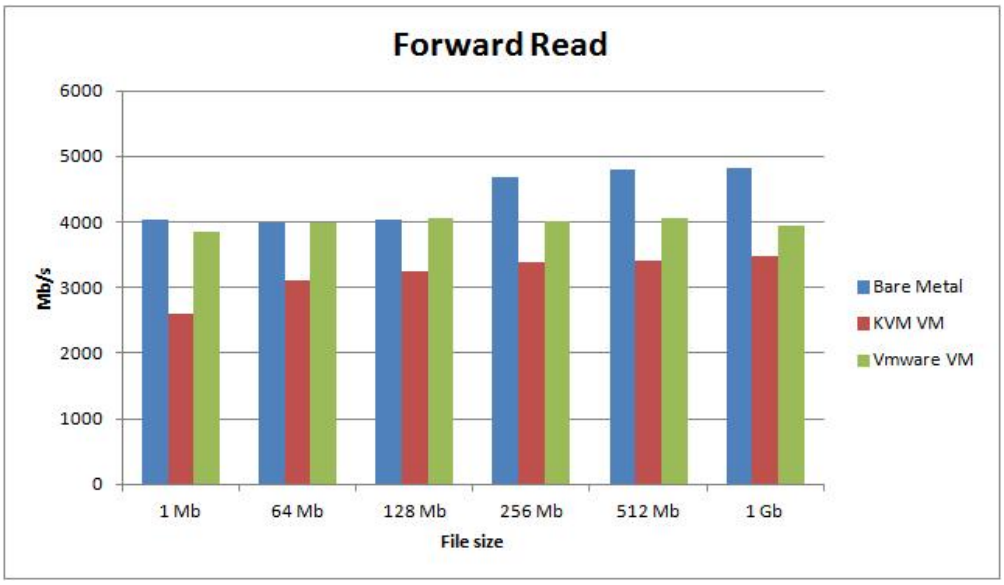

Figure 5.12: Iozone average forward read

5.1.13 Re-Forward Read

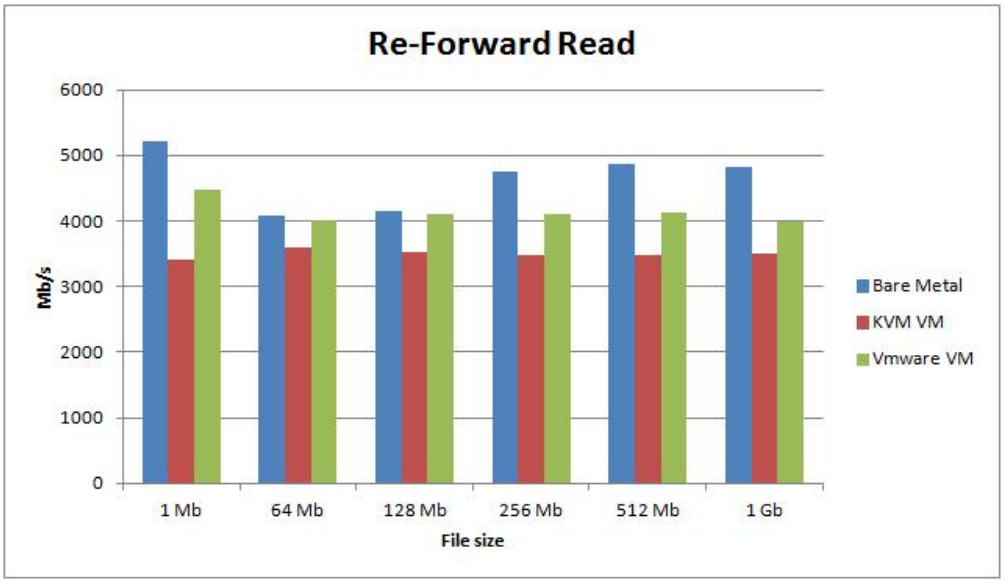

Figure 5.13: Iozone average re-forward read

5.2 Discussion of Iozone test results

Iozone measure the I/O with the help of 13 different kind of tests. The results from Bare Metal has shown better results as compared with KVM and VMWare. Virtual-

ization add overhead due to which virtualization guest performance remains lower as compared with Bare Metal. There is room for the improvement of virtualization technologies to reduce the overhead for better performance.

In most of the cases as illustrated from Iozone graphs, VMWare performed better than KVM. In some cases VMWare were found very close to Bare Metal system such as forward read 64 MB and 128 MB. Few instances was found interesting specially in case of 64 MB with write, rewrite, random write, record rewrite, forward write and forward re-write where KVM performed better than VMWare. It would be interesting to know the performance of KVM with file size smaller than 64 MB.

The worst performance of the KVM was observed in case of large file sizes. In case of 1 GB file size, in almost all the cases KVM performance was three time poorer than VMWare. KVM performance is better in case of smaller file size, but VMWare was close to Bare Metal guest. Better performance of the KVM was observed in case of all kind of reading.

5.3 Consolidated Iozone results

All kind of test performed by Iozone is either write or read. The write tests includes write, re-write, random write, record rewrite, forward write and re-forward write whereas read tests are composite of read, re-read, random read, backward read, stride read, forward read, re-forward read. The results of all the write and read tests are consolidated with regards to different file sizes.

5.3.1 Consolidated write performance

The write performance of Iozone test was consolidated and the percentage of KVM and VMWare was calculated in terms of Bare Metal. Sum up of all the write tests of Iozone was added and their percentage was calculated. The calculation of the percentage of write for KVM and VMWare was obtained by considering the Bare Metal as a base. All the figures given in the table 5.1 is in %.

Table 5.1: Consolidated Write performance of Iozone test

File size	KVM	VMWare
1 MB	61.0	91.7
64 MB	77.4	56.6
128 MB	74.0	96.3
256 MB	67.9	89.2
512 MB	65.5	86.1
1 GB	33.5	84.9

The consolidated graph for Iozone write test 5.14 presented in terms of percentage as compared with Bare Metal.

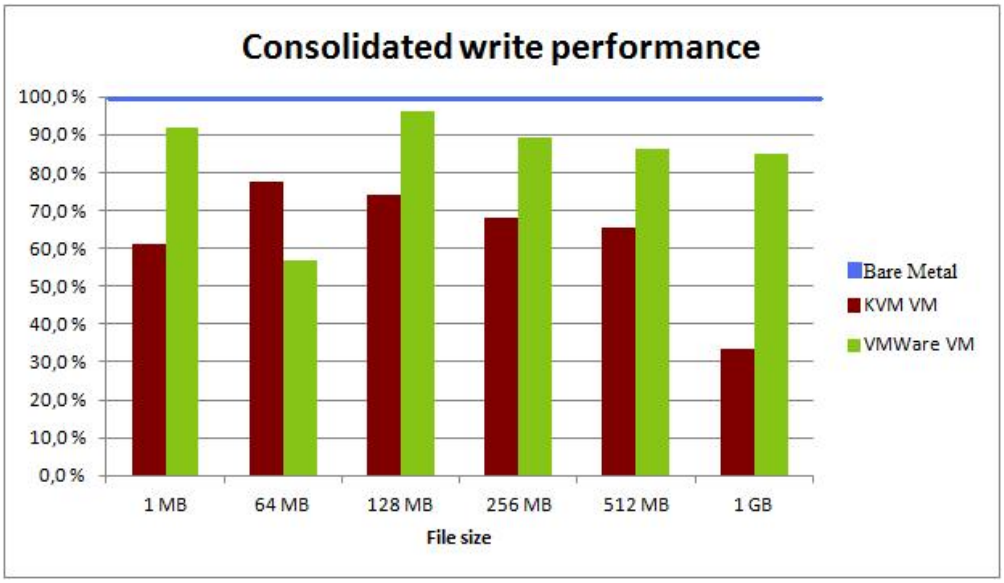

Figure 5.14: Consolidated write performance

The consolidate write performance 5.14 for Iozone test has shown the comparison of KVM and VMWare with Bare Metal. The optimum performance of Bare Metal has shown in blue line which is always 100%. VMWare has shown better performance in almost all the file size as compared with KVM. In case of 64 MB file size KVM has shown 78% performance and VMware was at 57% when compared with Bare Metal 100%. The performance of VMWare in case of 1 GB file size was 85% which was more than twice fast while compared with KVM. In case of 128 MB file size VMWare shows best performance and just behind 2% when comparing with Bare Metal.

5.3.2 Consolidated read performance

The consolidated read performance of Iozone test for KVM and VMWare was calculated in terms of Bare Metal. All the read tests of Iozone were added and their percentage was calculated. The calculation of percentage for KVM and VMWare was obtained by considering the Bare Metal as a base. The Bare Metal figures considered 100% for KVM and VMWare. All the figures given in the table 5.2 is in %.

Table 5.2: Consolidated Read performance of Iozone test

File size	KVM	VMWare
1 MB	65.5	91.1
64 MB	81.2	100
128 MB	79.2	100
256 MB	69.9	85.8
512 MB	67.8	85.3
1 GB	69.0	83.4

The graph of consolidated read test of Iozone 5.15 is self explanatory:

Figure 5.15: Consolidated read performance

The consolidate read performance of Iozone 5.15 has shown that VMWare out performed the KVM in all file sizes. In file size 64 MB and 128 MB, VMWare performance was identical to Bare Metal system. While rest of the cases VMWare performance was better than KVM. In 1 MB file size KVM has shown 65% performance while compared with Bare Metal.

5.4 Ram speed

In this section graphs for ram speed are presented. For data reliability the ram speed was repeated 25 time with the help of script. Detail of data summary and Box plot for Bare Metal, KVM and VMWare included as **Appendix B**. Ram speed was tested with exponential of 2 KB block size with maximum of 2 GB. Average from all the test results was calculated and presented in the following graphs.

5.4.1 Integer and Writing

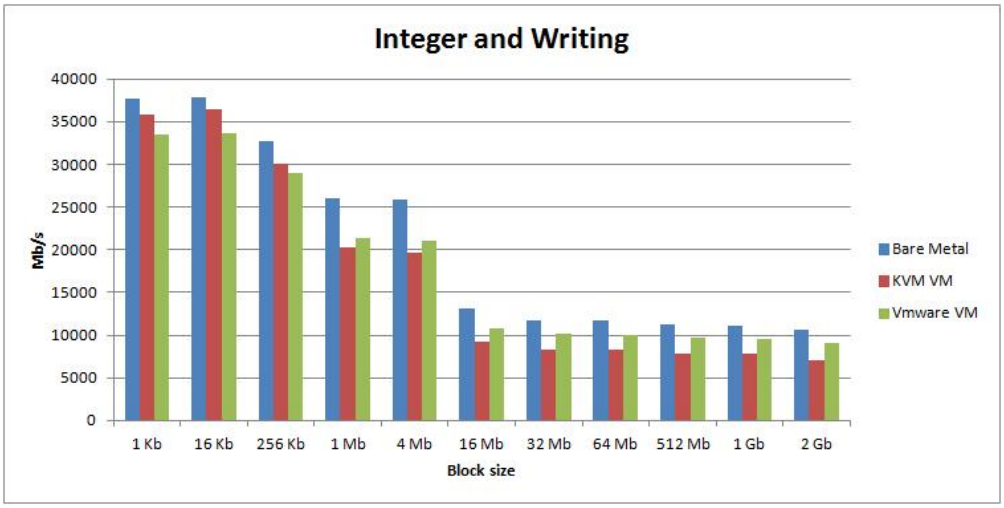

Figure 5.16: Ram speed average integer and writing

5.4.2 Integer and Reading

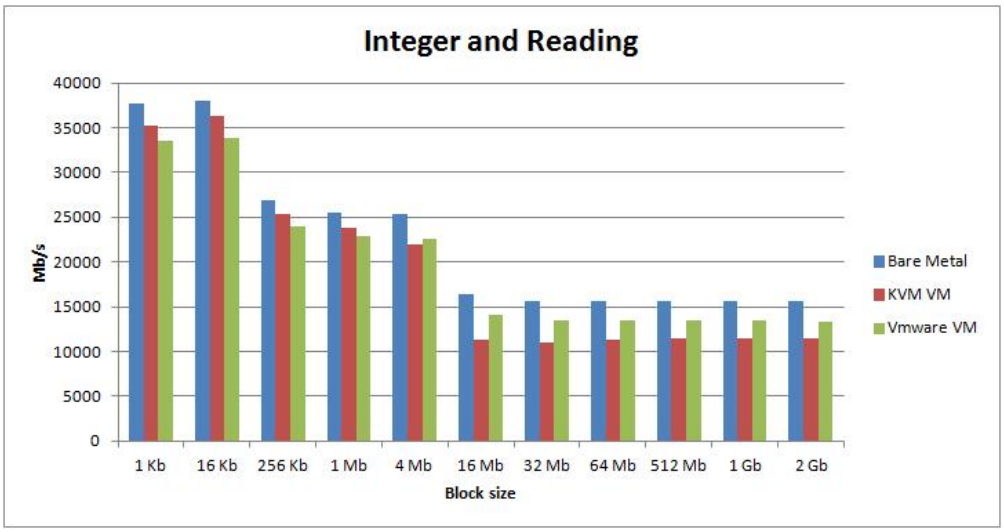

Figure 5.17: Ram speed average integer and reading

5.4.3 Float and Writing

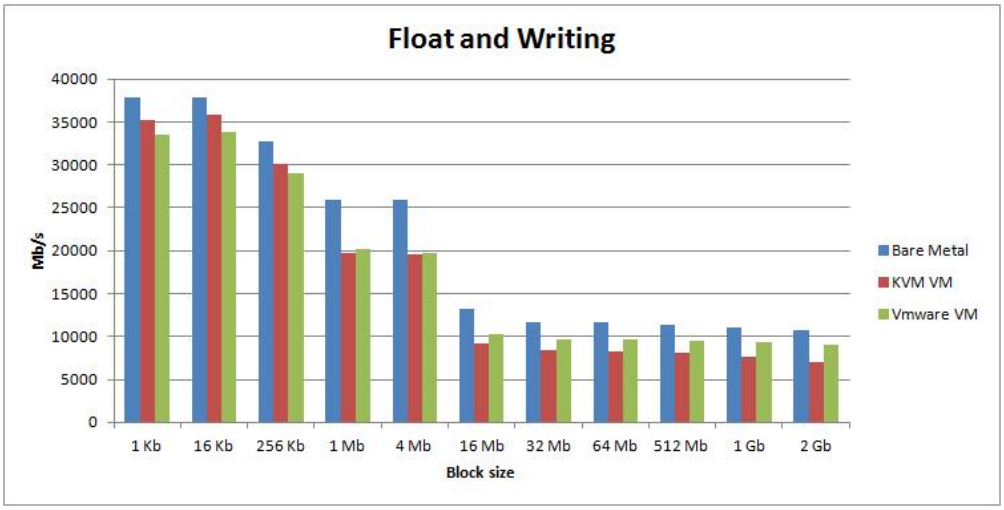

Figure 5.18: Ram speed average float and writing

5.4.4 Float and Reading

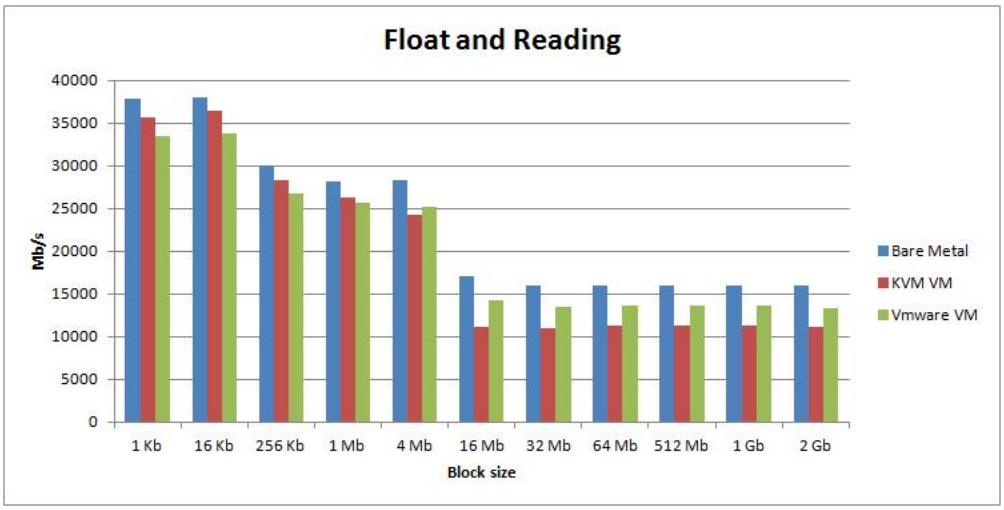

Figure 5.19: Ram speed average float and reading

5.5 Discussion of Ram Speed test results

Ram speed measures the memory performance by using four different kind of tests that include Integer reading and writing, Float reading and writing. Ram speed uses the block size for performance measurement. In all the cases of ram speed test, Bare Metal has shown the best possible results as compared with KVM and VMWare. The abstraction layer used by the virtualization technologies caused the lower performance

of the KVM and VMWare when comparing with Bare Metal.

While compared the two virtualization systems, performance of both were different with different memory block sizes. In case of block sizes smaller than 1 MB, KVM perform better than VMWare. In case of block size larger than 1 MB, VMWare performance observed better than KVM. In case of float and writing 5.18 with 4 MB block size, KVM and VMWare performance was observed identical.

5.6 Consolidated Ram speed results

The four tests of the Ram speed are formatted with integer and float reading and writing. The consolidated percentage of writing and reading for KVM and VMWare was calculated. The addition of all the reads and writes was performed separately. Bare Metal considered as base for KVM and VMWare percentage calculation. The KVM and VMWare compared with respect to Bare Metal.

5.6.1 Consolidated performance of integer and float writing

The table 5.3 shows the performance of KVM and VMWare for writing while comparing with Bare Metal.

Table 5.3: Consolidated performance of integer and float writing

File size	KVM	VMWare
1 KB	94.0	88.5
16 KB	95.5	89.1
256 KB	91.9	88.4
1 MB	77.3	79.9
4 MB	75.4	78.7
16 MB	70.8	80.5
32 MB	70.7	84.5
64 MB	70.7	84.4
512 MB	70.2	85.2
1 GB	69.2	85.1
2 GB	66.4	84.8

The Bare Metal shows the optimum performance of 100% in blue on top of the graph 5.20

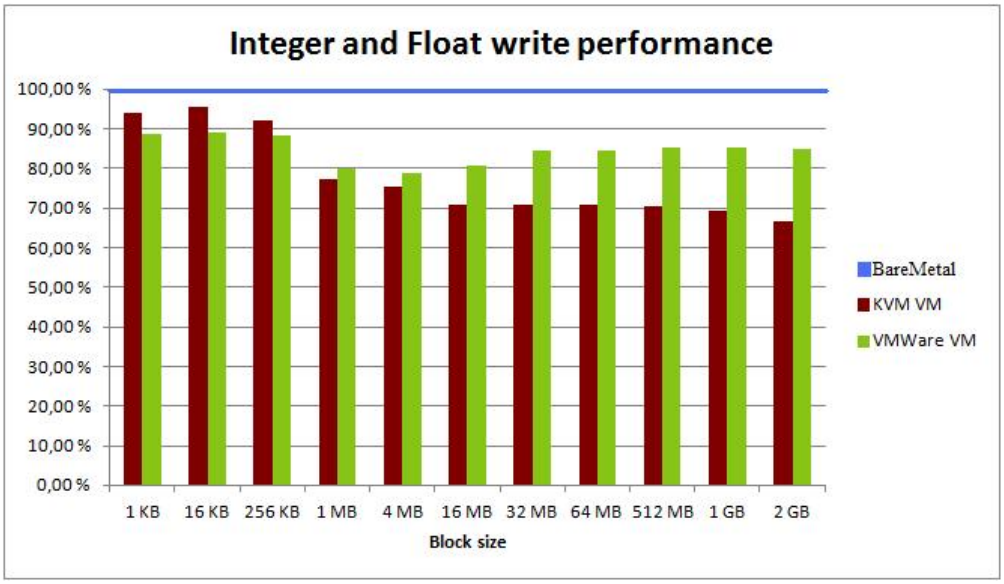

Figure 5.20: Integer and Float writing

The writing performance of KVM in smaller block sizes was observed better than VMWare and remained more than 93% of Bare Metal. Whereas the block size larger than 1 MB, the performance of the KVM remained roughly constant with average of 70% of Bare Metal. The performance of VMware was under 90% of Bare Metal in block size smaller then 1 MB. A continuous rised in performance of VMWare was observed from block size 1 MB and larger.

5.6.2 Consolidated performance of integer and float reading

The consolidated performance of memory with reading has shown in table 5.4. The performance of KVM and VMWare was calculated as percentage compared with Bare Metal.

Table 5.4: Consolidated performance of integer and float reading

File size	KVM	VMWare
1 KB	93.8	88.6
16 KB	95.8	89.1
256 KB	94.3	89.2
1 MB	93.6	90.4
4 MB	85.9	88.9
16 MB	67.1	84.7
32 MB	69.4	85.3
64 MB	71.6	85.7
512 MB	71.9	85.6
1 GB	72.3	85.7
2 GB	71.4	84.4

The performance of Bare Metal was maximum for KVM and VMWare and shown in blue line in the graph 5.21

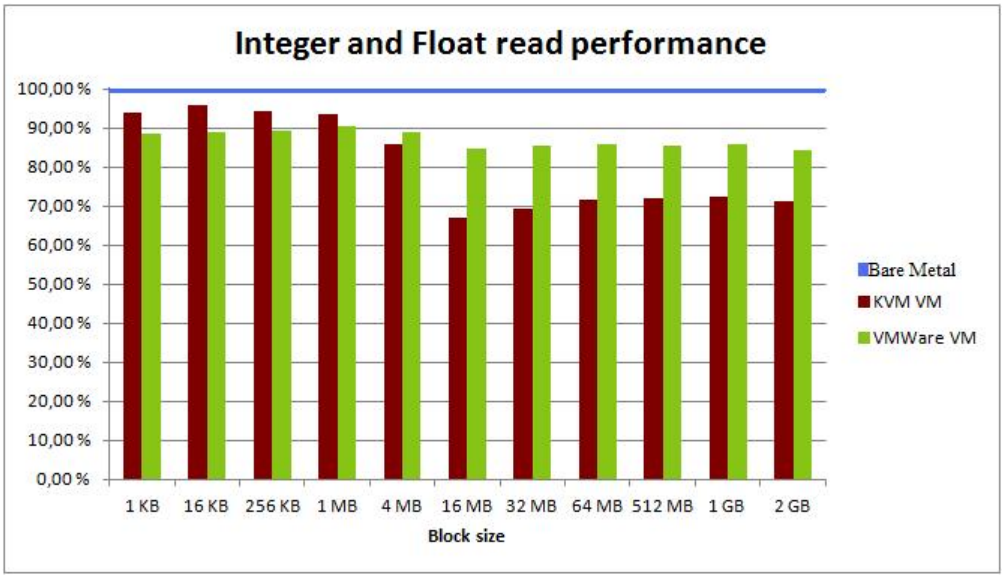

Figure 5.21: Integer and Float reading

In case of reading of the ram speed, the consolidated performance 5.21 of KVM was better when compared with VMWare in smaller block size. While in larger block sizes, the VMWare performance was remained 85% of the Bare Metal. KVM in larger block size was remained 70% of the Bare Metal.

5.7 Discussion

This section has been divided into different parts that includes spread of data with confidence interval and p-value.

5.7.1 Data spread

The data was collected after several test for each benchmark. With every run of the test new data collected that was little bit different from the previous test run. The variation in data was the reason for collecting the data for several runs. The box plot only gives the information about the placement of the data that which part of box contains the 25% and 75% of the data. It also mentions the maximum and minimum values of the data. Whereas the data spread shows the difference between the mean $\overline{X}$ and the standard deviation σ of the data. It was interesting to see that how much data is spread. The data spread range in terms of statistics can be calculate using standard deviation plus and minus mean of the data and denoted by:

$$\overline{X} + \sigma < \overline{X} < \overline{X} - \sigma$$

It was observed that data spread was larger in case of KVM and VMWare as compared

with Bare Metal guest. Followings box plot 5.22 is data spread example selected for analysis.

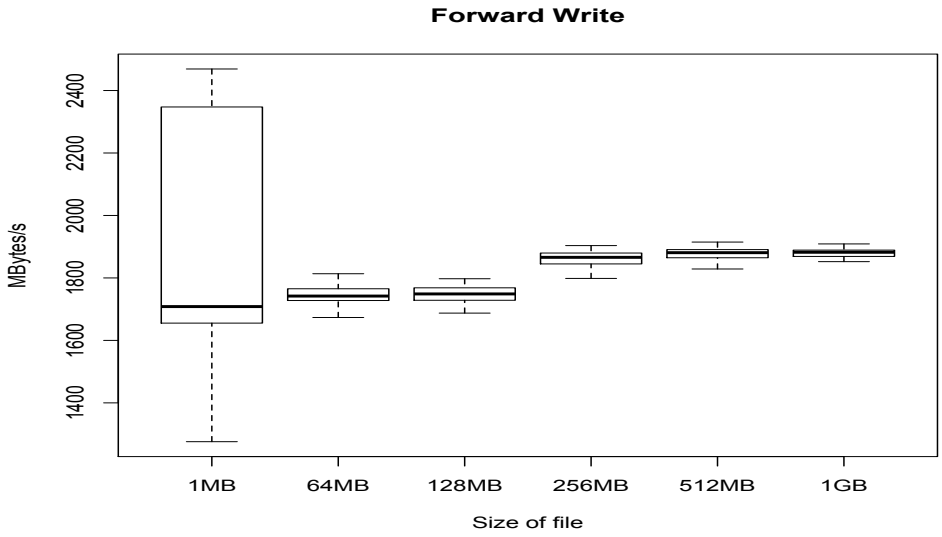

Figure 5.22: Iozone forward write for Bare Metal

Box plot of the Bare Metal forward write of Iozone 5.22 has shown that the data spread was quite larger in case of 1 MB file size as compared with file sizes greater than 1 MB. In case of the bigger file size, spread of data observed quite smaller.

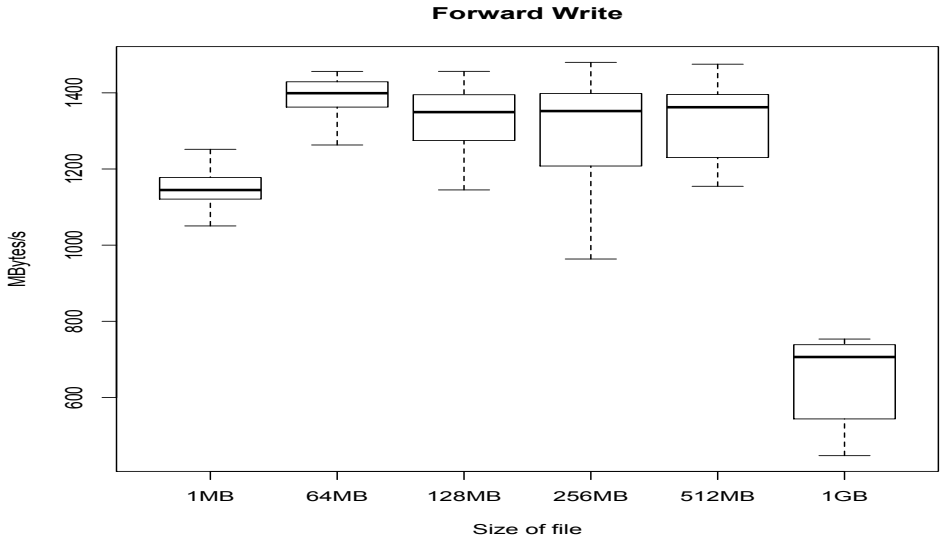

Figure 5.23: Iozone forward write for KVM

In case of KVM forward write of the Iozone 5.23, spread of the data was larger in large file size as compared with smaller file size.

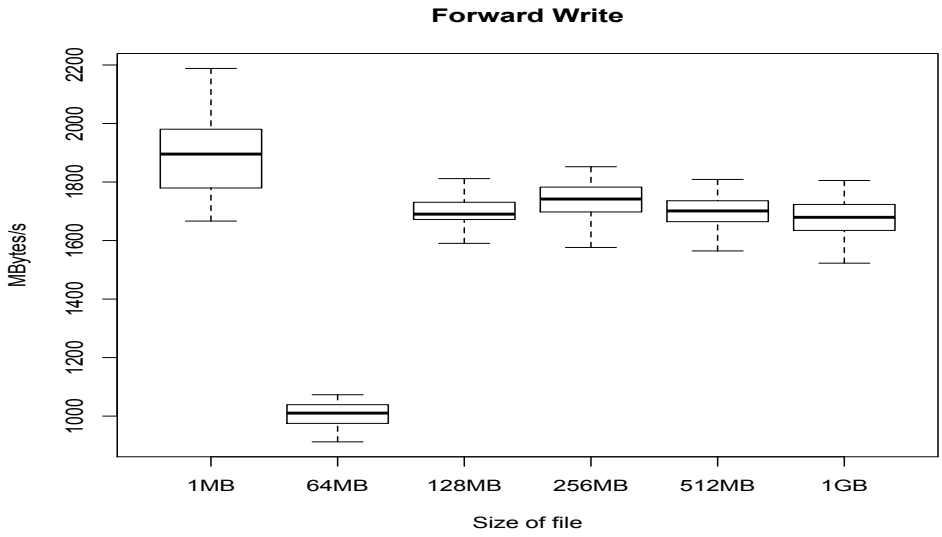

Figure 5.24: Iozone forward write for VMWare

The box plot 5.24 has shown that the data spread was also large in 1 MB file size as compared with bigger file sizes. Smaller spread of the data was observed where the file size was bigger than 1 MB in VMWare. In case of 1 MB file size Bare Metal data was more spread as compared with KVM and VMWare. The table 5.5 has shown the spread of data where the file size was 1 MB:

Table 5.5: Data spread of iozone forward write with 1 MB file size

	Bare Metal	KVM	VMWare
Mean($\overline{X}$)	1947.7	1141.6	1738.0
SD (σ)	344.4	164.1	385.2
$\overline{X}$ - σ	1603.3	977.5	1352.8
$\overline{X}$ + σ	2292.1	1305.7	2123.2

The data spread of the Bare Metal was almost 688.8 which was quite larger than 328.2 of the KVM. In case of VMWare the data spread was 770.4 which was more than Bare Metal and KVM. Another example of spread of data was included from the ram speed when testing the float and writing. Box plot for Bare Metal, KVM and VMWare has shown their data spread:

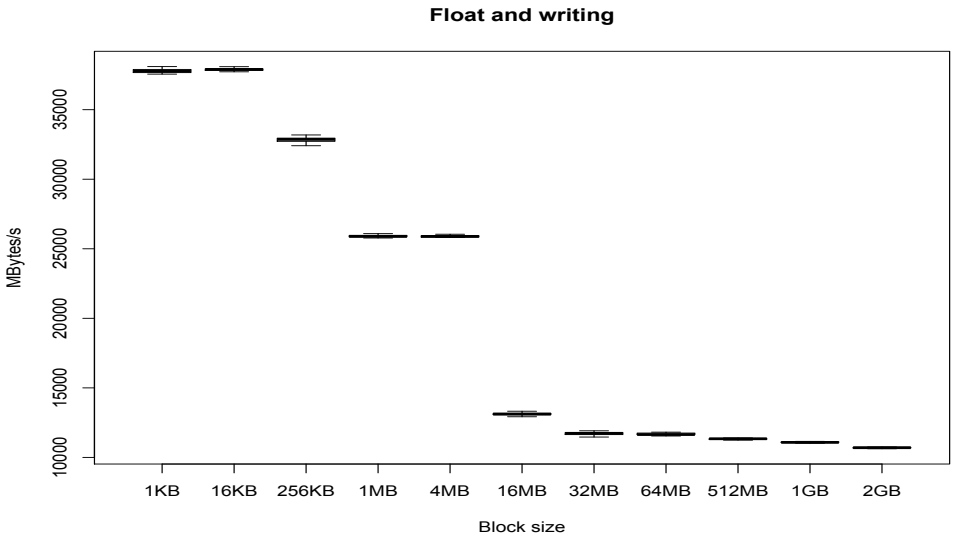

Figure 5.25: Ram speed float and writing for Bare Metal

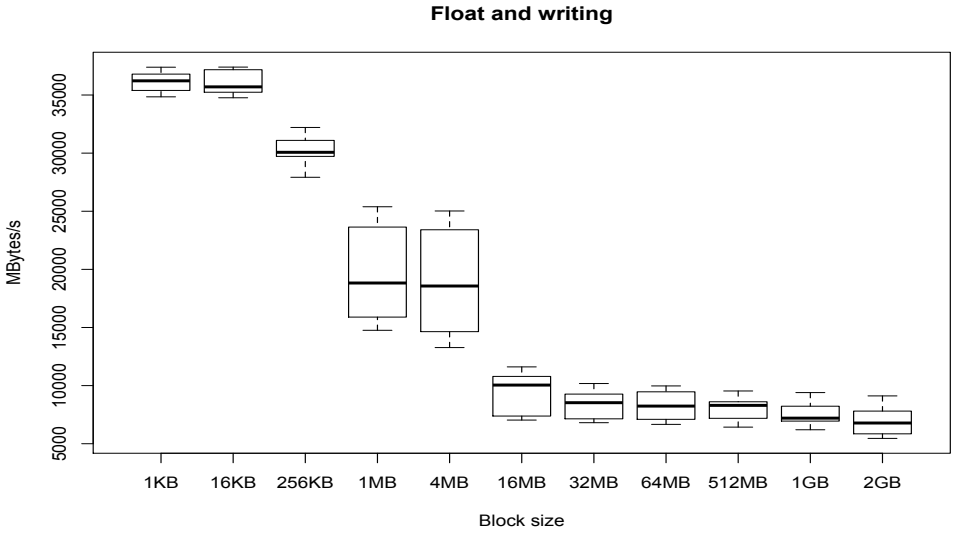

Figure 5.26: Ram speed float and writing for KVM

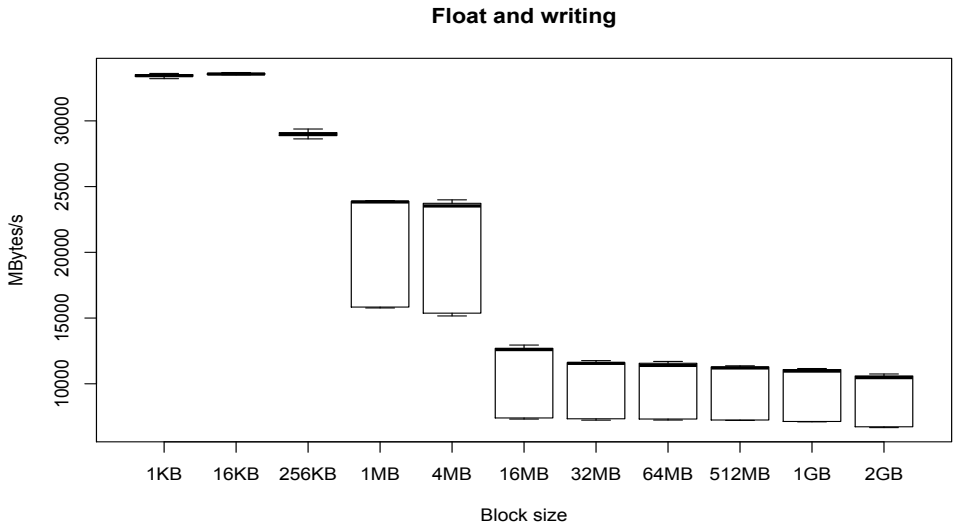

Figure 5.27: Ram speed float and writing for VMWare

As illustrated from the box plot, the data was less spread in case of the Bare Metal 5.25 as compared with the KVM 5.26 and VMWare 5.27. In case of VMWare most of the data was found in lower part of the box. In KVM the data spread was observed less where the block size was greater than 16 MB. The table 5.6 has shown the range of spread of data along with standard deviation in case of 4 MB block size:

Table 5.6: Data spread of ram speed for float and writing with 4 MB block size

	Bare Metal	KVM	VMWare
Mean($\overline{X}$)	25892.8	19503.0	19741.8
SD (σ)	94.1	4597.8	4252.2
$\overline{X}$ - σ	25798.7	14905.2	15489.6
$\overline{X}$ + σ	25986.9	24100.8	23994.0

From the table 5.6 , Bare Metal have only the spread of data 188.2. Whereas KVM and VMWare has a larger spread with 9195.6 and 8504.4 respectively. From the selected cases it was observed that the Bare Metal have a less data spread while compared with KVM and VMWare. Extreme difference in Bare Metal was observed as compared with KVM and VMWare in Ram Speed of data spread.

5.7.2 Confidence Interval

The data collected was aggregated after the number of runs of the benchmarks. To check the reliability of the mean estimation of the data collected, confidence interval was calculated for few selected cases. In first case where VMWare Read of Iozone with 1 MB file size was selected for confidence interval test. In order to find out how

accurate the test results, t-test was implemented. Results from t-test for VMWare with file size of 1 MB of Iozone are shown below:

T-test of 1 MB read of Iozone for VMWare

```
t.test(VMReader1mb)

        One Sample t-test

data:  VMReader1mb
t = 141.8315, df = 69, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
4025.562 4140.422
sample estimates:
mean of x
4082.992
Standard deviation 240.8546.
```

The N in this case was 70. The 95% confidence interval was

$$4025.5 < \overline{X} < 4140.4$$

The estimated $\overline{X}$ from this data was 4083.0. If the test repeated several time with N of 70, in 95% of the cases the mean will be within the range of the confidence interval. The confidence interval was very narrow with only difference of 114.9 as mentioned in figure 5.28.

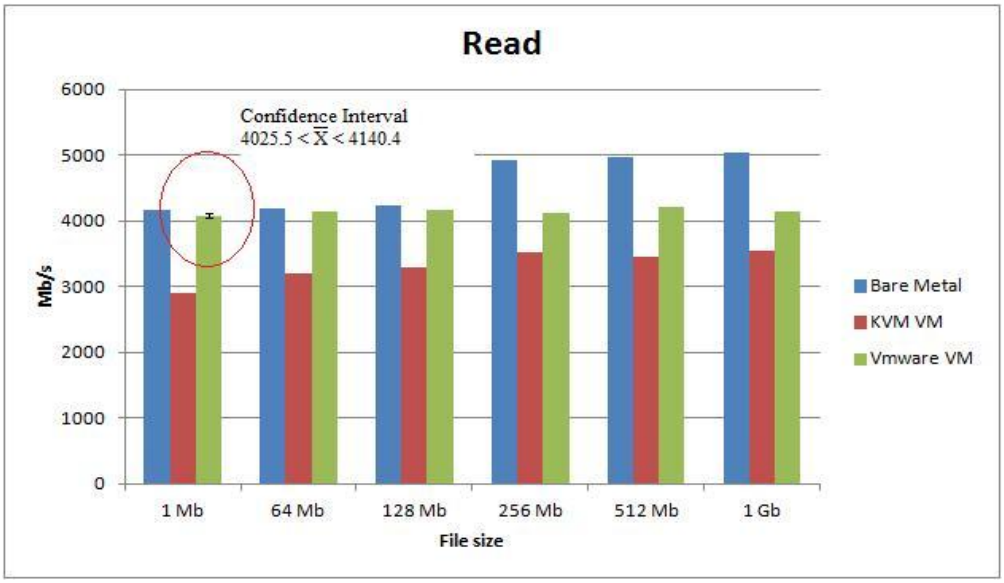

Figure 5.28: Confidence interval of iozone 1 MB file size of VMWare.

One more example for the confidence interval was taken from the Ram speed. Integer and reading with 4 MB of block size was selected from KVM to observe the confidence interval. In case of ram speed N was 25. The 95% confidence interval was

$$21503.2 < \overline{X} < 22313.5$$

The estimated $\overline{X}$ was remained 21908.4. In this case, 95% confidence that estimated mean in between range of confidence interval. The range of the confidence interval in this case was 810.3 as shown in figure 5.29.

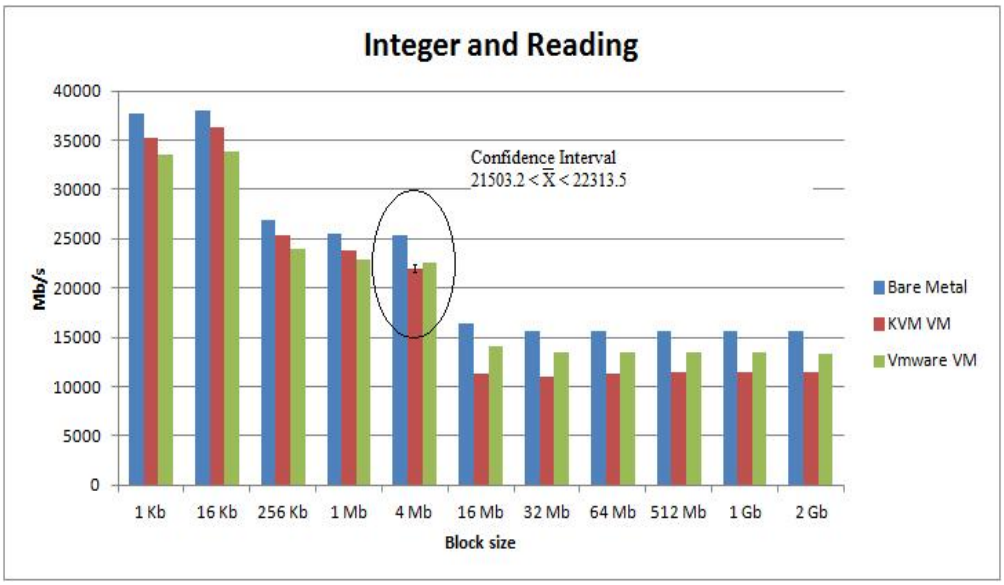

Figure 5.29: Confidence interval of ram speed 4 MB block size of KVM.

The results are considered as more accurate when the difference between the confidence interval remained very low. As in case of Iozone the difference of confidence interval was 114.9 which was very low and has shown the accuracy of the test results. The confidence interval was very narrow in most of the observations. That means, the difference is large enough to conclude that there is a significant difference between the estimated means. However, in a few cases the difference between the means is small and one should investigate this in detail by using an appropriate statistical test. The Welch T-test is implemented in the next section.

5.7.3 P-value

T-test was used to know the significant difference between the data samples. Few selected cases was taken and analyzed for this purpose to generalize the results. There are three different sample and t-test was used in following three ways.

- Bare Metal Vs KVM
- Bare Metal Vs VMWare
- KVM Vs VMWare

The p-value of the t-test between two samples has indicated the probability of being wrong in concluding that there is a difference in the two sample group. The out put of Welch two sample t-test generated from R is given below:

T-test of 4 MB Float and Writing of Ram speed for VMWare

```
> t.test(b44m,vmb44m)

        Welch Two Sample t-test

data:  b44m and vmb44m
t = 7.231, df = 24.023, p-value = 1.789e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 4395.484 7906.598
sample estimates:
mean of x mean of y
 25892.83  19741.79
```

If the P value :

$$P - value < 0.05$$

It can be concluded that there is significant difference between two sample groups. P value smaller than 0.05% indicates that 95% confidence about data occurrence between the interval.

Few cases were selected to know the p-value.

Table 5.7: P-value of t-test for comparison

	Bare Metal Vs KVM	Bare Metal Vs VMWare	KVM Vs VMWare
Iozone: Forward write with 64 MB file size	2.2^{-16}	2.2^{-16}	2.2^{-16}
Iozone: Random write with 64 MB file size	2.2^{-16}	2.2^{-16}	2.2^{-16}
Iozone: Read with 1 MB file size	2.2^{-16}	0.22	2.2^{-16}
Iozone: Backward Read 128 MB file size	2.2^{-16}	1.15^{-14}	2.2^{-16}
Iozone: Re-write 128 MB file size	2.2^{-16}	0.54	2.2^{-16}
Ram speed: Integer and reading with 1 KB block size	1.27^{-08}	2.2^{-16}	2.15^{-05}
Ram speed: Integer and reading with 4 MB block size	2.2^{-15}	2.2^{-16}	0.0015
Ram speed: Float and writing with 4 MB block size	3.47^{-07}	1.79^{-07}	0.85

In above table of the t-test, p-value between different sample groups were extremely low. If the p-value is extremely low, as in most the cases, means that there is significant different between the two population and it is not by chance. In case of Iozone

backward read with 128 MB, VMWare has shown better performance then Bare Metal. Bare Metal was on 3543 MB per second and VMWare was at 3767 MB per second. While compared both using t-test, p value was 0.54 which is greater than 0.05. In case of 1 MB read the p-value between Bare Metal and VMWare was 0.22 which is larger than 0.05.

In case of ram speed with float and writing with 4 MB block size, p -value between the KVM and VMWare was 0.85. This case was also shown that there was no significant difference between the two samples. In case of integer and reading with 4 MB block size, the p-value between KVM and VMWare was 0.0015 and lower than 0.05. It means that the difference is not statistically significant.

In most of the cases, p value has remained extremely low that indicated the accuracy of the results. On the bases of the above sample taken form the population, the results can be generalized for rest of the data collected. When the visual difference is not very small as in the cases discussed in confidence interval, we may say that there is significant difference between the two data. When the difference is very small, than difference might not be statistically significant.

5.8 UnixBench

UnixBench test system's CPU performance through different kind of tests 3.2. All these tests are categorized for performance calculation that includes CPU throughput, inter process communication throughput and file system thourghput. Dhrystone 2, Whetstone, and Excel throughput are CPU performance benchmarking. Pipe throughput, process creation, Shell script (8 concurrent), System call overhead are performance measurement of inter process communications. Whereas File copy 256, 1024 and 4096 are file system benchmarking tools. Detail table of calculation of Bare Metal, KVM VM and VMWare VM is included in **Appendix C** . The performance of CPU throughput, Interprocess communication throughput and file system throughput was compared with the help of graphs.

5.8.1 CPU Throughput results

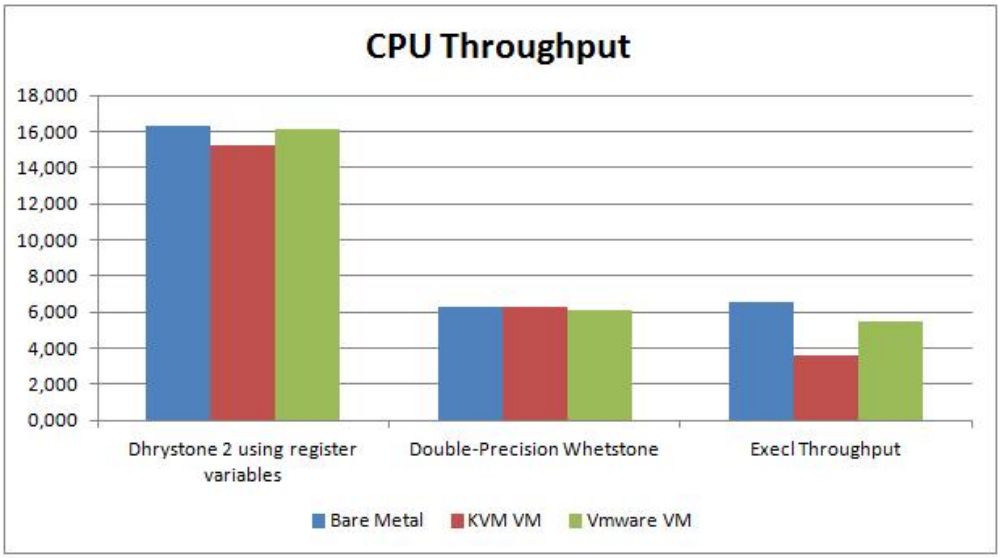

Figure 5.30: UnixBench CPU Throughput

The CPU throughput performance has shown in graph 5.30. Performance of Bare Metal was remained on the top followed by VMWare that was near to Bare Metal system. Whereas performance of KVM remained little behind than both Bare Metal and VMWare. Throughput score remained lower for the following tests:

- *Dhrystone 2* test, evaluates the manipulation of arrays, character strings, indirect addressing, and other common non floating point instructions. Bare Metal just over 16, KVM 15 and VMWare shows 16 iterations per seconds.
- *Double-precision Whetstone* is arithmetic test, and evalutes assignment, addition, subtraction and multiplication calculations. Bare Metal and KVM was shown 6 whereas VMWare remained below 6 iterations per seconds.

- *Execl Throughput* test the replacement of a currently running process with a new process. In this test Bare Metal remains over 6 and VMWare remains under 6. Whereas KVM was remains under 4.

In case of ALU , the performance of the KVM remained similar to Bare Metal, whereas the VMWare remained 2% behind to the Bare Metal. The ALU performance of all the three guests was remained Performance of the virtualization technologies remain lower than that of Bare Metal in all the cases of CPU performance test. The abstraction layer of virtualization caused of this performance decrement.

5.8.2 Inter Process Communication results

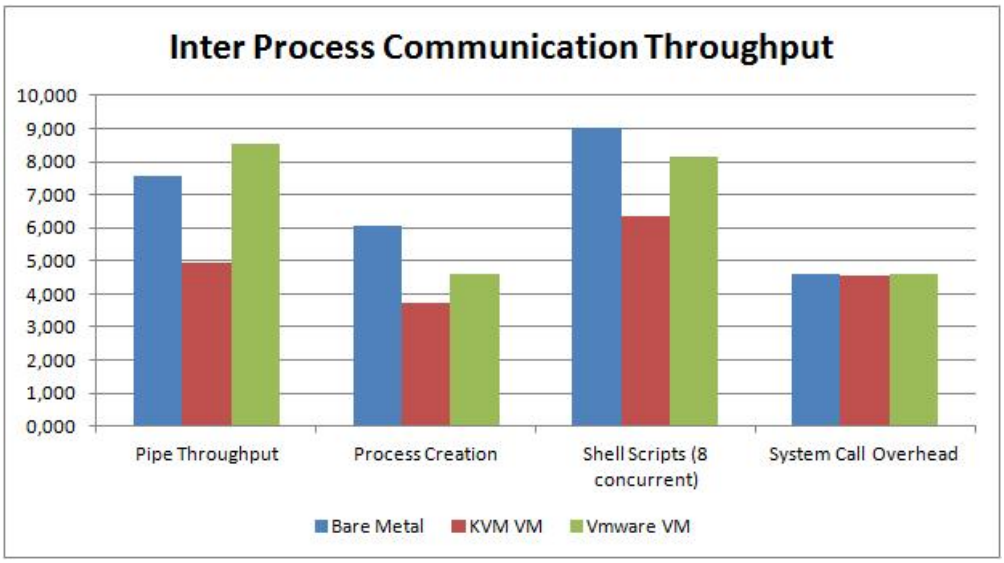

Figure 5.31: UnixBench Inter process Communication Throughput

In inter process communication 5.31 test both the virtualized systems produced the following results as compared with Bare Metal:

- *Pipe Throughput* evaluates a single process which opens a pipe to itself and communicates data in loop. Bare Metal shows 7.5 and KVM remains just under 5 iteration per second. Whereas VMWare shows 8.5, that was quite unusual. The reason might be at the time of Bare Metal performance test CPU was busy with other process or due to some unknown reasons.

- *Process Creation* evaluates the repeated creation of a child process which immediately dies after its own fork(). Performance of Bare Metal was 6 as compared with KVM just 4 and VMWare over 4 iterations per second.

- *Shell Scripts* evaluates a shell script that is run by 1, 2, 4, and 8 concurrent processes. Iterations per second was remain at 9, 8 and 6.3 for Bare Metal, VMWare and KVM respectably.

- *System Call Overhead* evaluates the time required to do iterations of dup(), close(), getpid(), getuid(), and umask() calls. A marginal difference was observed in all the three cases and performance was around 4.5 iteration per second.

In case of system call overhead, the performance of KVM and VMWare was remained identical to the Bare Metal system. In case of pipe throughput, VMWare has shown even better performance than Bare Metal that might be because of some errors. VMWare produced better performance results as compared with KVM. However, both the virtualization technologies yet behind the Bare Metal. Virtualization overhead was very visible effect on performance and there was room for improvement for both approaches.

5.8.3 File System Throughput results

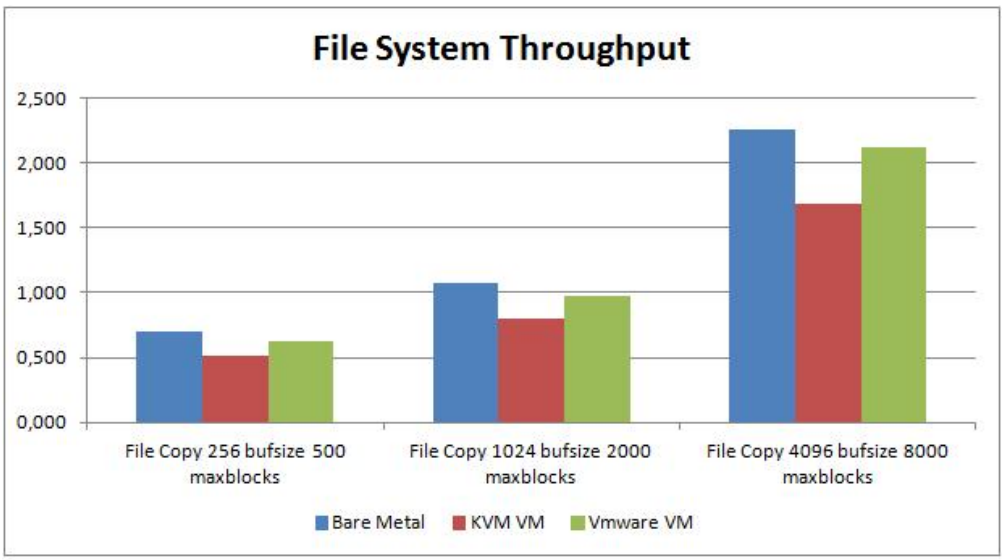

Figure 5.32: UnixBench File System Throughput

The file system 5.32 test measures the number of characters that can be copied within 10 second depending upon buffer sizes of 256 bytes, 1 KB and 4 KB. Bare Metal has shown better performance than KVM and VMWare as reflected in following results.

- *File copy 256 bytes* In case of Bare Metal, it was 0.7. Whereas VMWare remains at 0.06 and KVM at last with 0.05 iteration per second.
- *File copy 1 KB* Iteration per second was remain 1.100, 0.900 and 0.80 for Bare Metal, VMWare and KVM respectably.
- *File copy 4 KB* Bare Metal has shown the highest performance of 2.3 followed by VMWare with 2.1 iteration per second. Whereas KVM has shown 1.6 iteration per second remained the lowest.

Again VMWare performance better than KVM and very close to Bare Metal. KVM added more overhead in file system throughput and is needed improvement in this area for better results.

5.8.4 Composite Throughput score

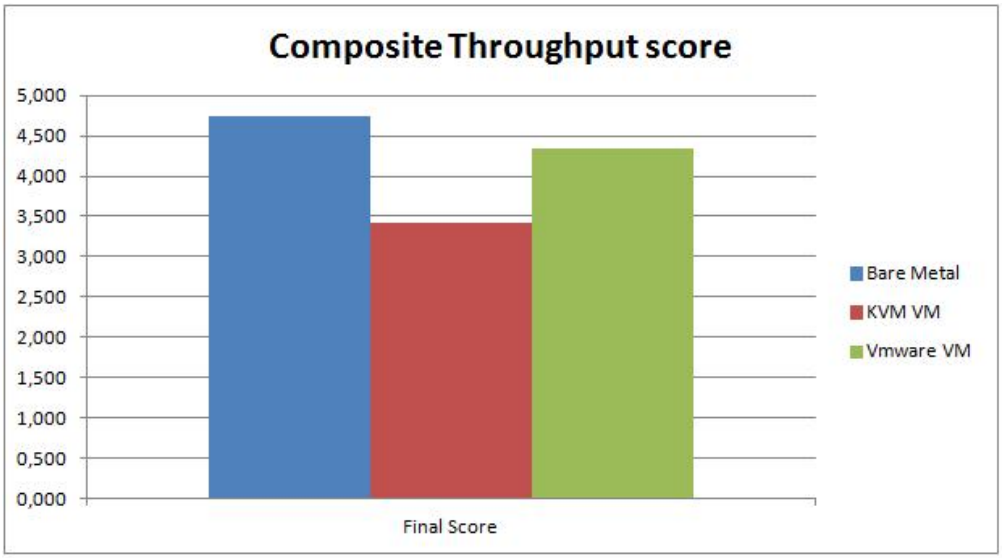

Figure 5.33: UnixBench composite throughput score

CPU composite throughput performance 5.33 has shown that Bare Metal was at top performance with highest score of 4.7. Whereas both the virtualized technologies remained 4.4 and 3.7 in case of VMWare and KVM respectively.

The UnixBench tests focus on different system resources including CPU, file systems, pipes and processes. These processes communicate with system kernel services and activate kernel-level memory events. All the benchmark test in UnixBench uses aggregate timing for performance measurement and for this purpose it uses shell command time. The abstraction layer added by virtualization in different technology have different effects, some are using hypervisor while other are using virtual machine monitor. Difference in architecture effects the difference in performance. In system composite performance VMWare far ahead than KVM and near to Bare Metal guest system.

5.9 Future work

Iozone have maximum of 1 GB file size for test. However, it would be interested to benchmark the guests with larger file than 1 GB. Similar with Ram speed, that have maximum of 2 GB block size to test the guest. By using the larger block size, some interesting facts cab be unfold. The performance comparison of the KVM and VMWare was made by using Iozone for I/O, ram speed for memory and UnixBench for

cpu. It would also be interested to compare the performance for KVM and VMWare with some other available benchmarking tools. This research was conducted by using the RHEL 6.1. This would also be interesting to compare the performance of the KVM and VMWare by using the different operating systems. The interesting facts can also be unfolded to measure the performance of the KVM and VMWare hypervisors by running several virtual machines. This can help to measures the scalability of the KVM and VMWare.

Chapter 6

Conclusions

Virtualization nowadays is very popular technology, as it reduce the management cost, expenses and many more benefits for the organizations. Many virtualization technologies are available in the market. The selection of the right technology for the organization can produce better results. KVM virtualization is gaining popularity whereas VMWare virtualization is the market leader. Most of the organizations are using VMWare.

The primary purpose of this research work was to make the comparison between KVM and VMWare. The secondary purpose was to compare the performance of virtualized and non virtualized guests. In the case of a virtualized environment, the abstraction layer between hardware resources and OS, is obviously affecting the performance of the virtual guest. KVM and VMWare are both different technologies for virtualization and using different architectures. KVM uses OS layer or para virtualization approach whereas the VMWare uses the hardware layer virtualization. This different approaches of virtualization might have created the difference in performance.

While comparing the non-virtualized and virtualized environment, it was observed that the Bare Metal out perform the KVM and VMWare in almost every test. When comparing KVM and VMWare, some interesting results were observed. In case of Iozone, when writing large files, VMWare was more than twice as fast as KVM. In the special case of writing 64 MB files it was observed that KVM was more than 30% better than VMWare. While in reading the VMWare perform better than KVM. In case of smaller file sizes VMWare was 20 to 25% better than KVM. In special case of 1 MB file, VMWare was 40% better than KVM.

While testing the memory performance with ram speed with block size smaller than 4 MB, KVM perform 4 to 7% better than VMWare. In case of block size larger than 4 MB, VMWare perform 15 to 25% better than KVM. In case of other writing VMWare observed 30 to 50% better than KVM. CPU performance was measured by using UnixBench. The performance of VMWare was quite close to the performance of Bare Metal. In case of ALU bound processes, no overhead was observed and performance KVM and VMWare was almost similar to Bare Metal. In UnixBench composite throughput score, Bare Metal perform 9% better than VMWare and 29%

better an KVM. Whereas VMWare perform 20% better than KVM.

In overall performance, the VMWare perform better than KVM. In some cases VMWare performs twice better than KVM. Whereas. In few cases KVM also gave better results than VMWare.

Bibliography

[1] Steven J. Vaughan-Nichols, *New Approach to Virtualization Is a Lightweight* IEEE Computer Society, pp. 12-14, November, 2006.

[2] Herrod, S. A., *Systems Research and Development at VMware* ACM SIGOPS Operating Systems Review archive, Vol. 44(4), ACM New York, NY, USA, December 2010.

[3] Crosby, S. and Brown, D., *The Virtualization Reality.* ACM QUEUE Jr., pp.34-41, December/January 2006-2007

[4] Goth, G., *Virtualization: Old Technology Offers Huge New Potential.* Published by the IEEE Computer Society, Vol. 8(2), February 2007.

[5] Hirt, T., *KVM - The kernel-based virtual machine.* February 2010.

[6] Oguchi, Y. and Yamamoto, T. Server, *Virtualization Technology and Its Latest Trends* FUJITSU Sci. Tech. J, Vol. 44(1), pp.46-52, January 2008.

[7] Singh, A. *An Introduction to Virtualization.* http://www.kernelthread.com/publications/virtualization/ Accessed on, 17 January 2011 [20:00], 2004.

[8] Sander, P. and Bobo, S., *The 100 Best Technology Stocks you can buy 2012.* Aadams Media Inc. USA, pp.337-339, January 2012.

[9] VMWare Inc, *VMware ESXi and ESX Info Center.* http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html?src=WWW_ESXIInfoCenter_US_HPBanner_ESXIMigration#utm_source=WWW_ESXIInfoCenter_US_HPBanner_ESXIMigration&utm_medium=src&utm_campaign=src-tagged-url Accessed on, 15 February 2012.

[10] Guan, T., Hai, J., Xia, X., Wenzhi, C. and Pingpeng, Y., *Measuring and Analyzing CPU Overhead of Virtualization System.* Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific, pp.243-250, December 2011.

[11] Smith, J. E. and Nair, R., *Virtual Machines: versatile platforms for systems and processes.* Morgan Kaufmann publishers, May 2005.

[12] Neiger, G., Amy, S., Leing, F., Rodgers, D., and Uhlig, R., *Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization.* Intel Technology Journal, Vol. 10(03), 2006.

[13] Apparao, P., Iyer, R., Zhang, X., Newell, D., and Adelmeyer, T., *Characterization and Analysis of a Server Consolidation Benchmark.* In Proceedings of ACM/USENIX International Conference on Virtual Execution Environments (VEE), 2008.

[14] Lee, B. and Brooks, D., *Accurate and efficient regression modeling for microarchitectural performance and power prediction.* In Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, pp.185-194, New York, NY, USA, 2006.

[15] Hauswirth, M., Diwan, A., Sweeney, P., and Mozer, M., *Automating vertical profiling.* In Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications, pp.281-296, New York, NY,USA, 2005.

[16] Chiueh, S. N. T, and Brook, S., *A Survey on Virtualization Technologies.* RPE Report, pp. 1-42, 2005.

[17] Smith, J. E., *The architecture of virtual machines.* Computer, Vol. 38(5), pp. 32-38, May 2005.

[18] Marinescu, D. and Kroger, R., *State of the art in autonomic computing and virtualization.* Distributed Systems Lab, Wiesbaden University of Applied Sciences, Wiesbaden, Germany, September, 2007.

[19] Goldberg, Robert P., *Survey of Virtual Machine Research.* IEEE Computer Magazine, Vol. 7, pp. 34-45, June, 1974.

[20] Popek, G. J. and Goldberg, R. P., *Formal requirements for virtualizable third generation architectures.* Commun. ACM, Vol. 17, pp. 412-421, July, 1974.

[21] Sugerman, J., Venkitachalam, G., and Lim, B. H., *Virtualizing I/O Devices on VMware Workstations Hosted Virtual Machine Monitor.* Proceedings of the General Track: 2002 USENIX Annual Technical Conference. USENIX Association, pp. 1-14, 2002.

[22] Soltesz, S., Poetzl, H., Fiuczynski, M., E., Bavier, A., and Peterson, L., *Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors.* Proceedings of the 2007 conference on EuroSys. ACM Press., pp. 275-287, 2007.

[23] Rosenblum, M. and Garfinkel, T., *Virtual machine monitors: current technology and future trends.* Computer, Vol. 38(5), pp. 39-47, 2005.

[24] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A., *Xen and the art of virtualization.* Proceedings of the 19th ACM symposium on Operating systems principles. ACM Press., pp. 164-177, 2005.

[25] Adams, K. and Agesen, O., *A comparison of software and hardware techniques for x86 virtualization.* Proceedings of the 12th international conference on Architectural support for programming languages and operating systems. ACM Press., pp. 2-13, 2006.

[26] Oppegaard, M. C. A., *Evaluation of Performance and Space Utilization When Using Snapshots in the ZFS and Hammer File Systems.* Master Thesis, Oslo University College, pp. 29-36, 2009.

[27] Cherkasova, L. and Gardner, R., *Measuring CPU overhead for I/O processing in the Xen virtual machine monitor.* In Proceedings of the USENIX Annual Technical Conference, p.24, Berkeley, CA, USA, 2005.

[28] Bray, T., *Bonnie.* http://www.textuality.com/bonnie/, 1996, Accessed 28 Feb, 2012.

[29] Coker, R., *Bonnie++.* http://www.coker.com.au/bonnie++/, 2001, Accessed 28 Feb, 2012.

[30] Katcher, J., *Postmark, a new file system benchmark.* Technical report, Network Appliance, Oct 1997.

[31] Traeger, A., Zadok, E., Joukov, N., and Wright, C. P., *A nine year study of file system and storage benchmarking.* Trans. Storage, Vol. 4(2), pp.1-56, 2008.

[32] Capps, D., *A nine year study of file system and storage benchmarking.* http://www.iozone.org/docs/IOzone-msword-98.pdf, Accessed 28 of Feb, 2012.

[33] Qumranet, *White Paper KVM Kernel based Virtualization Driver.* http://www.redhat.com/products/virtualization/, Accessed 09 of March, 2012.

[34] Walters, J. P., Chaudhary, V., Cha, M., Guercio Jr., S., and Gallo, S., *A Comparison of Virtualization Technologies for HPC.* 22nd International Conference on Advanced Information Networking and Applications, IEEE Computer Society. pp.861-868. 2008.

[35] Nanda, S., and Chiueh, T., *A Survey on Virtualization Technologies.* Stony Brooks, New York. p.32, 2005.

[36] Che, J., Shi, C., Yu, Y., and Lin, W., *A Synthetical Performance Evaluation of OpenVZ, Xen and KVM.* IEEE Asia-Pacific Services Computing Conference. IEEE Computer Society. pp. 587-594, 2010.

[37] Menon, A., Santos, J. R., Turner, Y., Janakiraman, G. J., and Zwaenepoel, W., *Diagnosing performance overheads in the xen virtual machine environment.* In Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments (VEE 05), USA: ACM New York, pp. 13-23, 2005.

[38] Whitaker, A, Shaw, M., and Gribble, S., *Scale and Performance in the Denali Isolation Kernel.* In Proceedings of Symposium on Operating Systems Design and Implementation(OSDI 02), pp. 195-209, 2002.

[39] Clark, B., Deshane, T., Dow, E., Evanchik, S., Finlayson, M., Herne, J., and Matthews, J., *Xen and the Art of Repeated Research.* In Proceedings of the 2004 USENIX Annual Technical Conference, pp. 135-144, 2004.

[40] Ongaro, D., Cox, A. L., and Rixner, S., *Scheduling I/O in Virtual Machine Monitor.* In ACM/USENIX International Conference on Virtual Execution Environments(VEE 08), pp. 1-10, 2008.

[41] Landmann, R., Cantrell, D., De Goede, D. and Masters, J., *Red Hat Enterprise Linux 6, Installation Guide.* http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/index.html, Accessed February 10, 2012.

[42] Landmann, R., Cantrell, D., De Goede, D. and Masters, J., *Red Hat Enterprise Linux 6, Installation Guide.* http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/sect-Virtualization-Installing_the_virtualization_packages-Installing_KVM_packages_on_an_existing_Red_Hat_Enterprise_Linux_system.html, Accessed March 10, 2012.

[43] Deshane, T., Shepherd, Z., Matthews, J., Ben-Yehuda, M., Shah, A., and Rao, B., *Quantitative Comparison of Xen and KVM.* Xen Summit., June 23-24, 2008.

[44] Younge, A. J., Henschel, R., Brown, J. T., Laszewski, G. V., Qiu, J., and Geoffrey, C., *Analysis of Virtualization Technologies for High Performance Computing Environments.* IEEE 4th International Conference on Cloud Computing, IEEE Computer Society, Indiana University, U.S.A. pp.9-16, 2011.

[45] Fuertes, W., Vergara, J. E. L., Pincha, J., Aules, H., Jacome, L., and Grijalva, M., *Analytical Expression to Predict the Overhead Produced by the VMware and Xen Virtualization Tools.* High Performance Computing and Networking research group, Escuela Politecnica Superior, Universidad Autonoma de Madrid, Spain. 2012.

[46] Casazza, J., Greenfield, M., and Shi, K., *Redefining Server Performance Characterization for Virtualization Benchmarking.* Intel®Technology Journal. Vol 10(03), August 10, 2006.

[47] Intel Software Network, *Measuring Performance of Applications on Virtualized Systems Under Test (SUTs).* Technical Report. October 2008.

[48] Fuertes, W., Vergara, J. E. L., *A quantitative comparison of virtual network environments based on performance measurements.* In Proc. 14th HP Software University Association Workshop, Munich, Germany, pp. 8-11, July 2007.

[49] Waldspurger, C., *Memory resource management in VMware ESX server.* In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, 2002.

[50] VMware Inc., *Understanding Full Virtualization, Paravirtualization, and Hardware Assist.* VMware Inc., white paper,

http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf Accessed March 15, 2012.

[51] VMware Inc., *Understanding Memory Overhead.* VMware ESXi 4.0 Installable and vCenter Server 4.0 Edition, http://pubs.vmware.com/vsp40_i/wwhelp/wwhimpl/js/html/wwhelp.htm#href=resmgmt/c_understanding_memory_overhead.html, Accessed March 15, 2012.

[52] Smith, Z., *Bandwidth: a memory bandwidth benchmark.* http://zsmith.co/bandwidth.html, Accessed on March 17, 2012.

[53] Horvath, A., *MBW memory bandwidth benchmark.* http://ahorvath.home.cern.ch/ahorvath/mbw/, July 2004, Accessed on March 17, 2012.

[54] Coffey, P., Beliveau, J., Mogre, N., and Harner, A., *Benchmarking the Amazon Elastic Compute Cloud (EC2).* Worcester Polytechnic Institute, March 9, 2011.

[55] Hollander, R. M., and Bolotoff P. V., *RAMspeed, a cache and memory benchmarking tool.* http://alasir.com/software/ramspeed/, November 2002, Accessed on March 17, 2012.

[56] Stefan-Radu, M., *CPU Free BenchMark (former CPUMark) 2.2.* http://www.softpedia.com/get/System/Benchmarks/CPUMark.shtml, April 2008, Accessed on March 19, 2012.

[57] UnixBench, *UnixBench : Know Your VPS Computation Power.* http://www.hostingformula.net/unixbench-know-your-vps-computation-power/, Accessed on March 19, 2012.

[58] Diskeeper, *Virtualization and Disk Performance.* Diskeeper corporation, 2006.

[59] UnixBench, *byte-unixbench: A Unix Benchmark Suite* http://code.google.com/p/byte-unixbench/, Accessed on April 10, 2012.

[60] Kopytov, A., *SysBench Manual.* http://sysbench.sourceforge.net/docs/, Accessed on April 12, 2012.

[61] Chambers, J., *The R Project for Statistical Computing.* http://www.r-project.org/, Accessed on May 16, 2012.

Appendix A

Iozone Results

A.1 Bare Metal Iozone results

The data summary that includes the minimum, lower and upper quartiles, median, mean and maximum was generated by using R software. The data summary was generated for all the data of Iozone and Ram speed. Table for data summary for Bare Metal write A.1 is presented as sample.

A.1.1 Write

Table A.1: Bare Metal Iozone Write

File size	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum
1 MB	602.4	750.3	766.5	867.6	1025	1079
64 MB	890.1	970.6	992.4	990.8	1016	1050
128 MB	943.1	994.8	1010	1009	1027	1049
256 MB	1088	1114	1122	1121	1128	1140
512 MB	1075	1124	1130	1129	1135	1144
1 GB	1108	1133	1137	1136	1140	1150

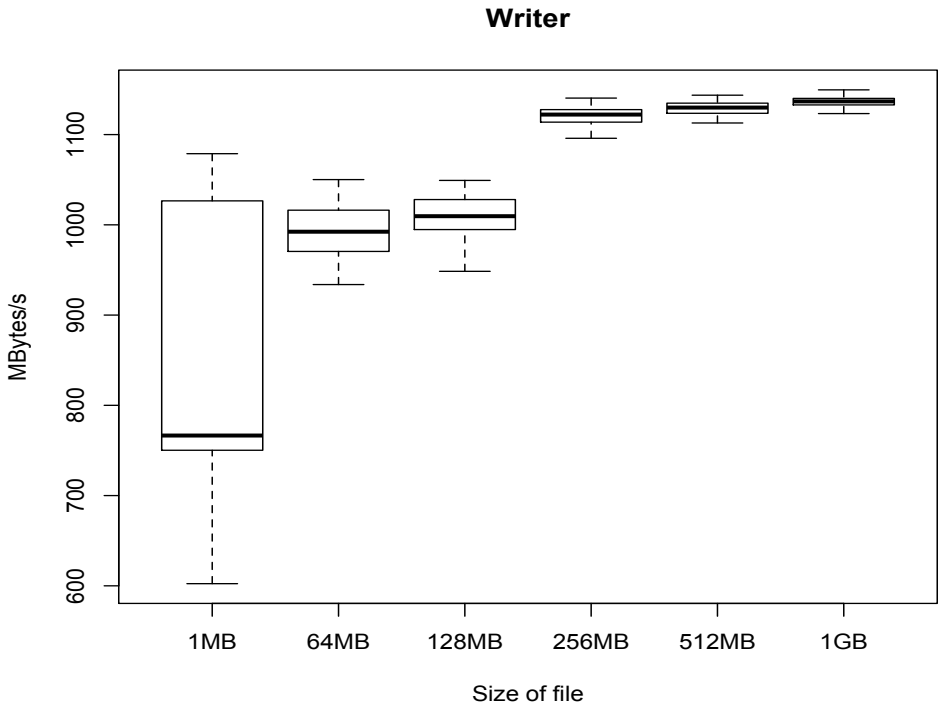

Figure A.1: Bare Metal Iozone Write

A.1.2 Re-Write

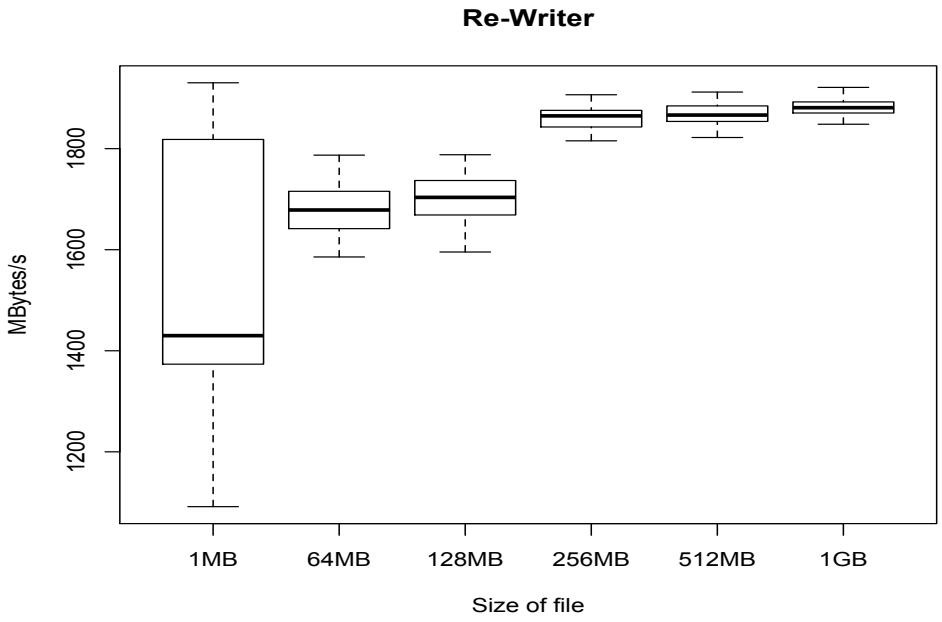

Figure A.2: Bare Metal Iozone Re-write

A.1.3 Read

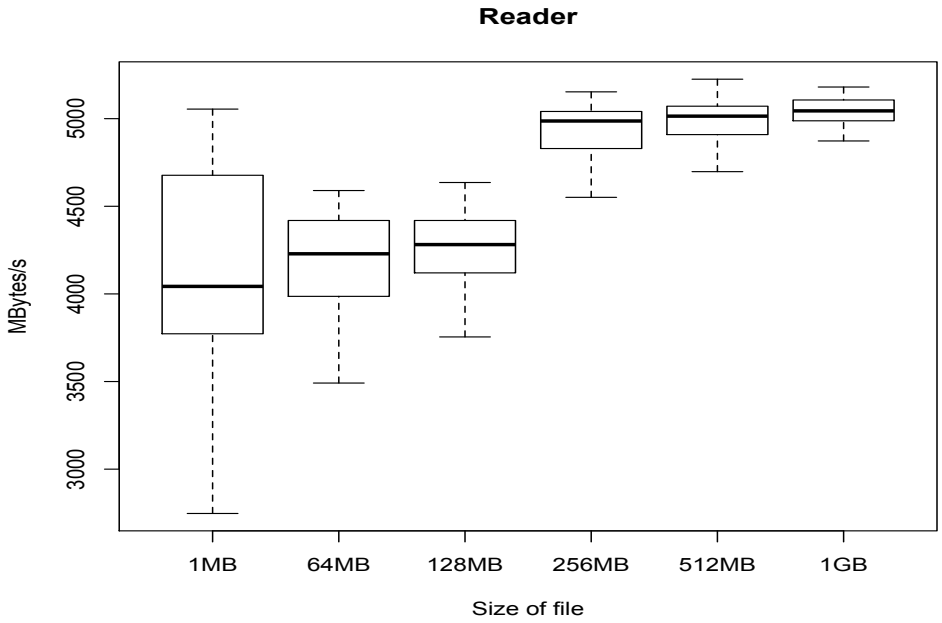

Figure A.3: Bare Metal Iozone Read

A.1.4 Re-Read

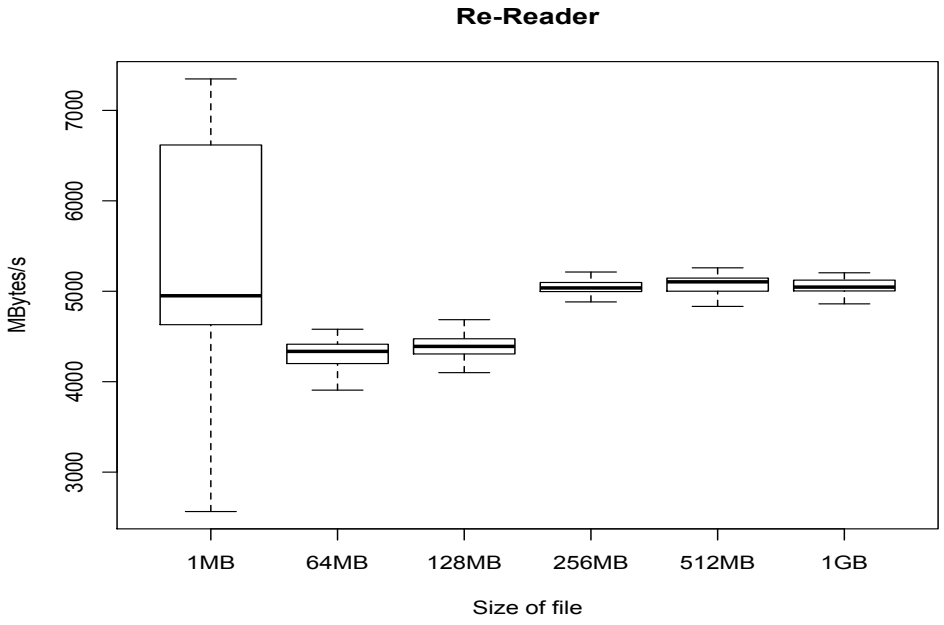

Figure A.4: Bare Metal Iozone Re-read

A.1.5 Random Read

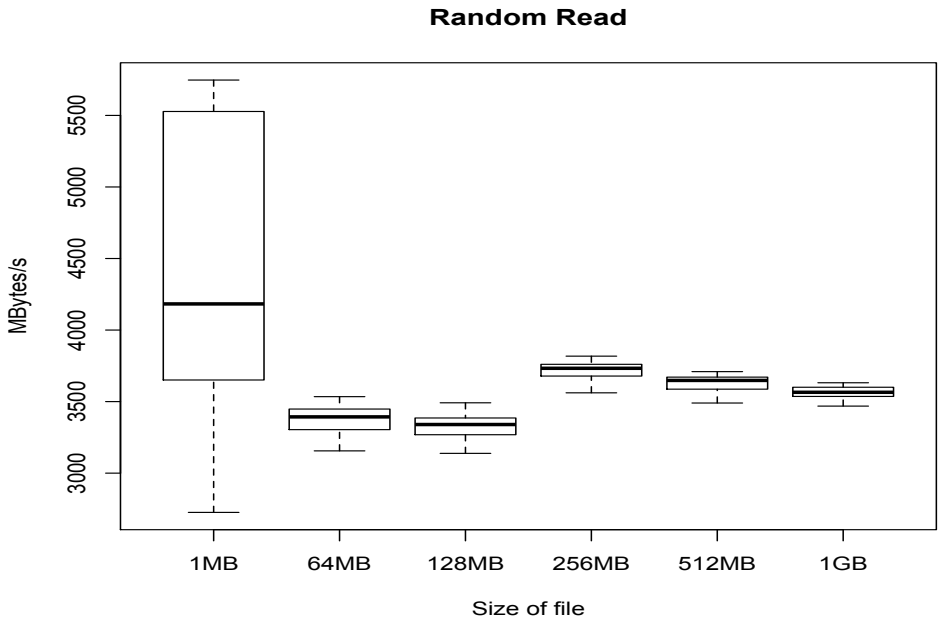

Figure A.5: Bare Metal Iozone Random read

A.1.6 Random Write

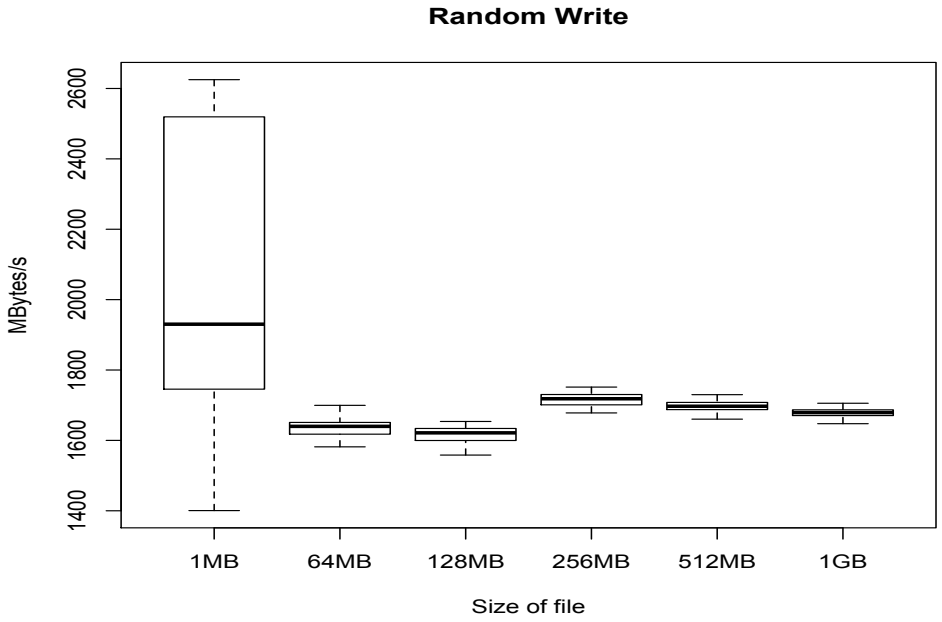

Figure A.6: Bare Metal Iozone Random write

A.1.7 Backward Read

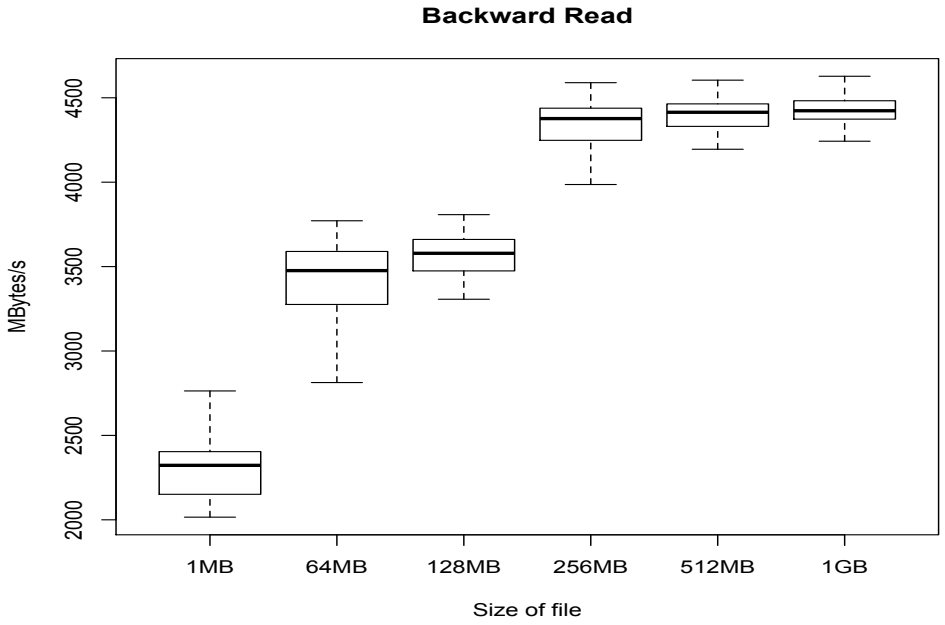

Figure A.7: Bare Metal Iozone Backward read

A.1.8 Record Re-write

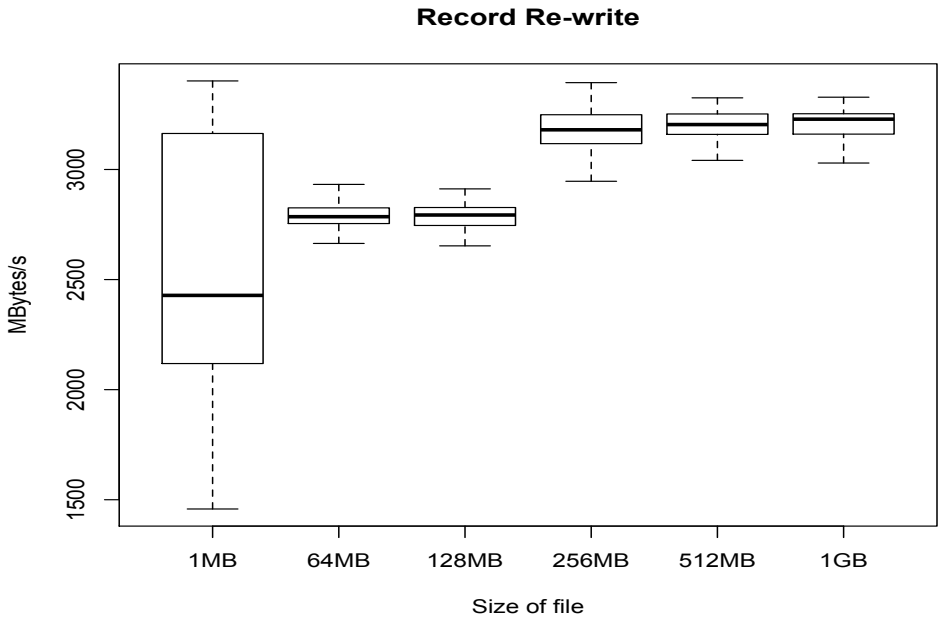

Figure A.8: Bare Metal Iozone Record re-write

A.1.9 Stride Read

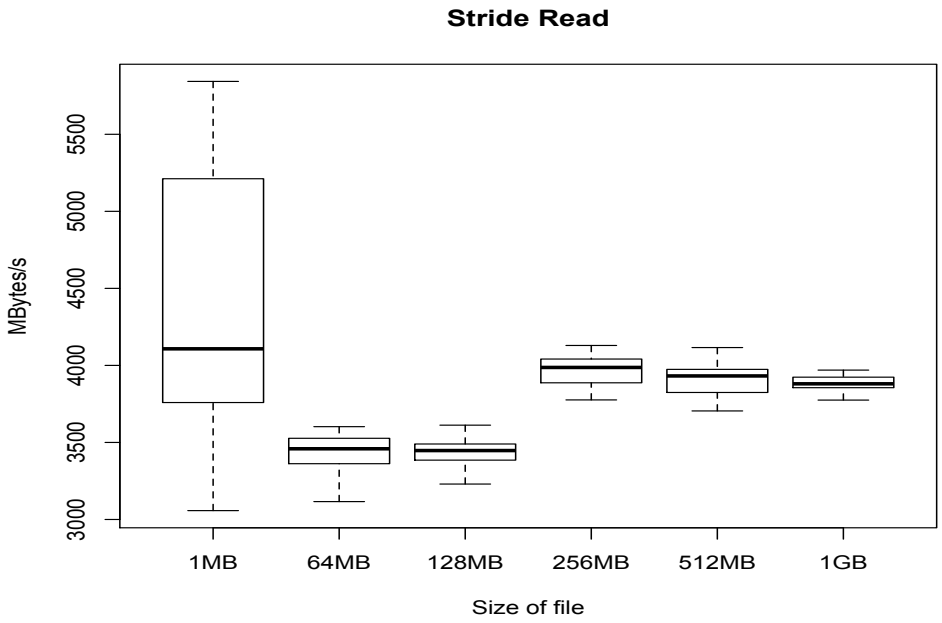

Figure A.9: Bare Metal Iozone Stride read

A.1.10 Forward Write

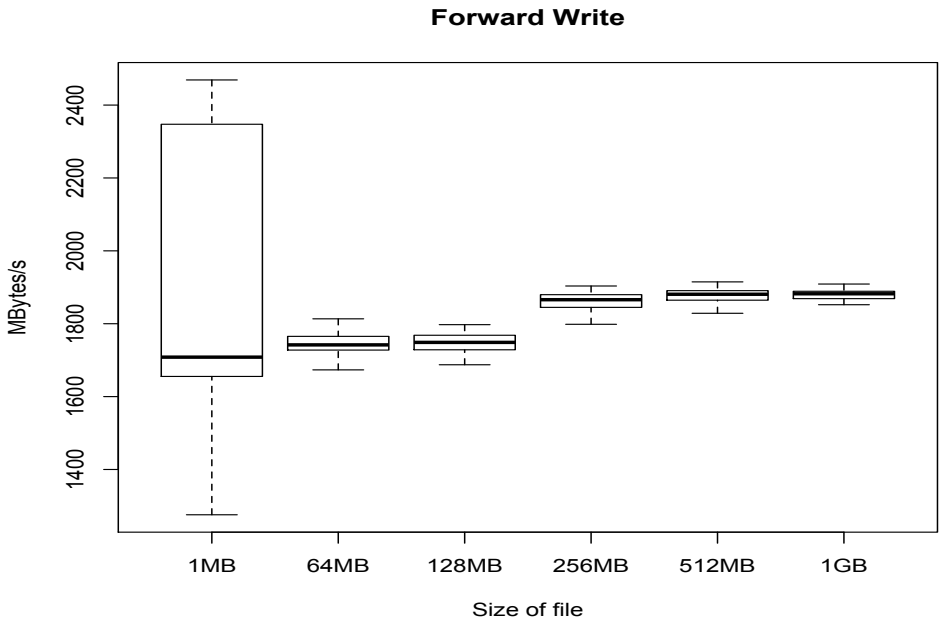

Figure A.10: Bare Metal Iozone Forward write

A.1.11 Re-Forward Write

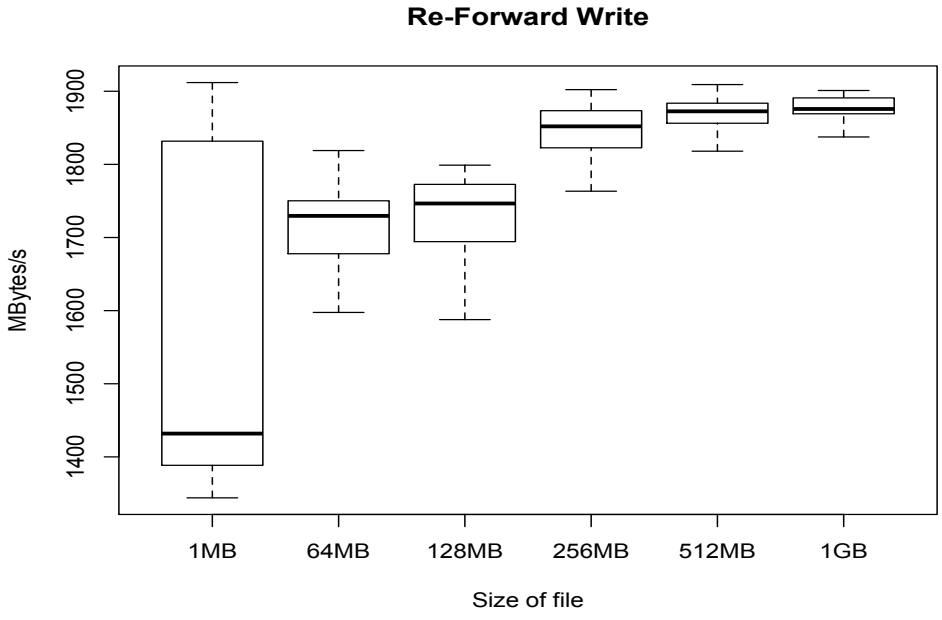

Figure A.11: Bare Metal Iozone Re-Forward write

A.1.12 Forward Read

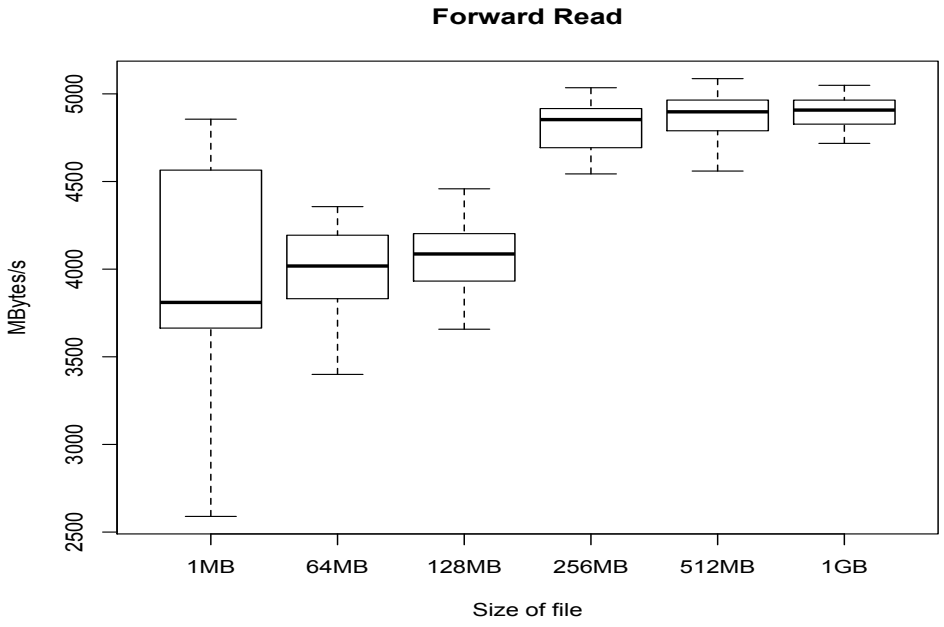

Figure A.12: Bare Metal Iozone Forward read

A.1.13 Re-Forward Read

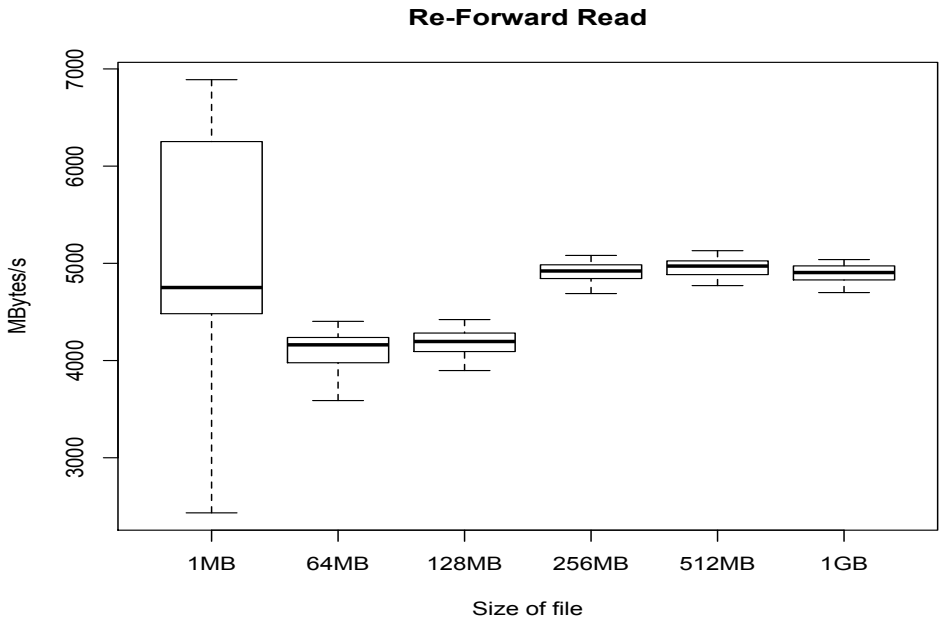

Figure A.13: Bare Metal Iozone Re-Forward read

A.2 KVM Virtual Machine Iozone results

A.2.1 Write

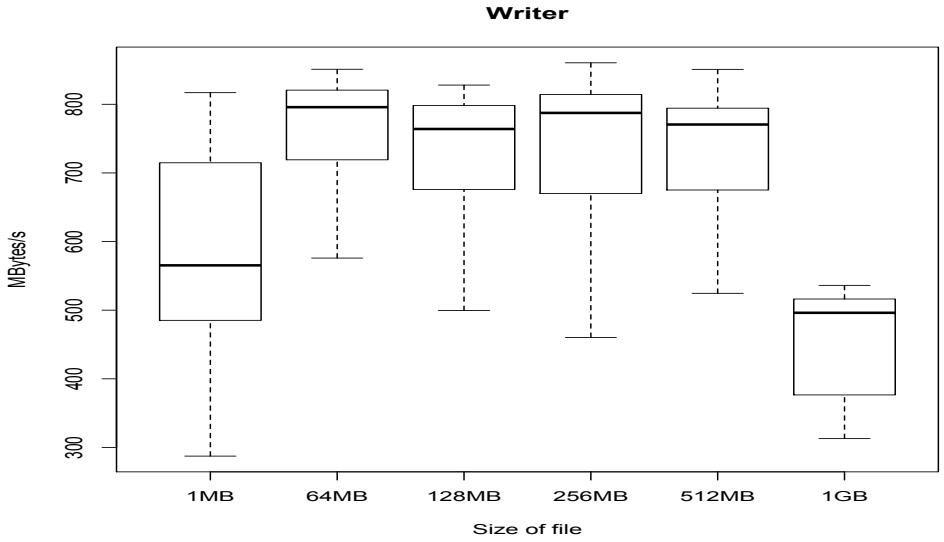

Figure A.14: KVM VM Iozone Write

A.2.2 Re-Write

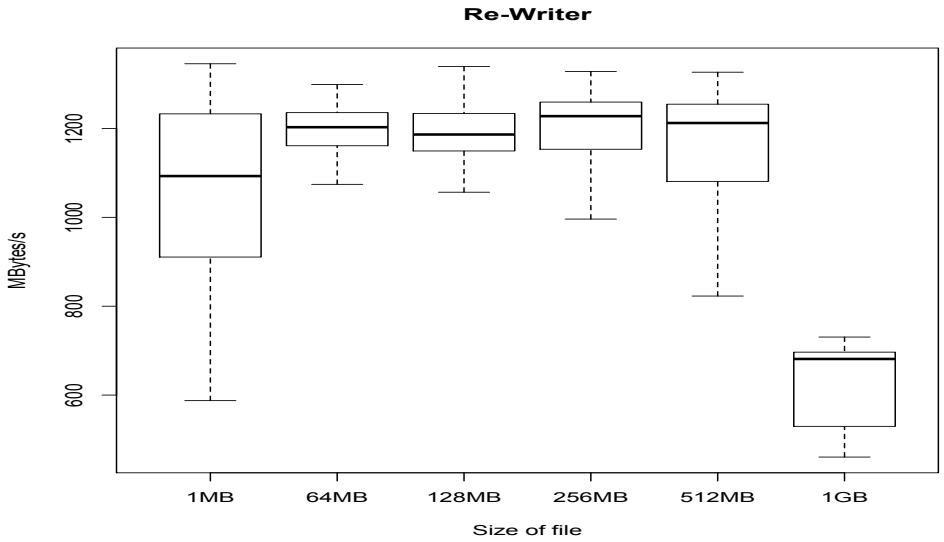

Figure A.15: KVM VM Iozone Re-write

A.2.3 Read

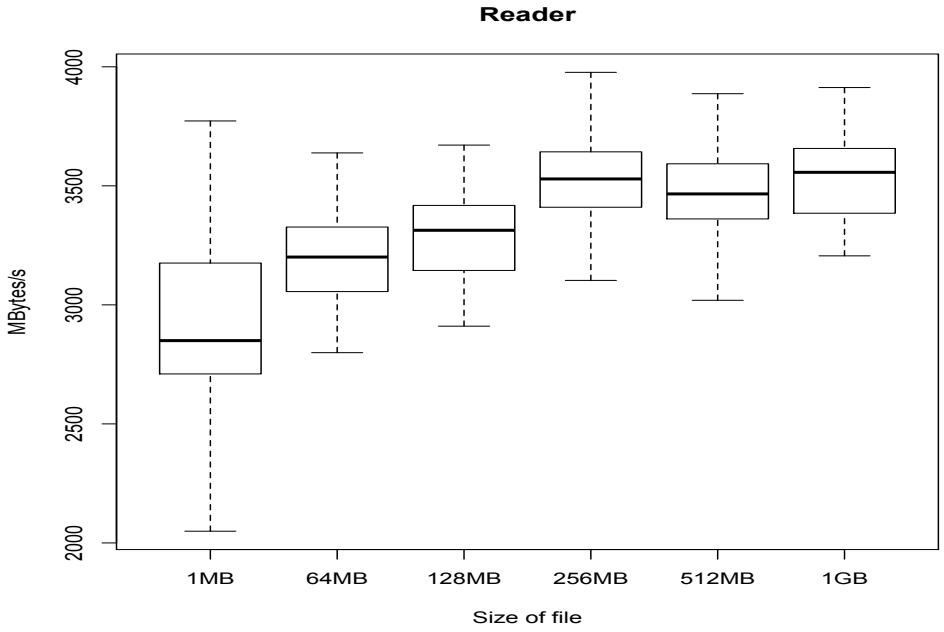

Figure A.16: KVM VM Iozone Read

A.2.4 Re-Read

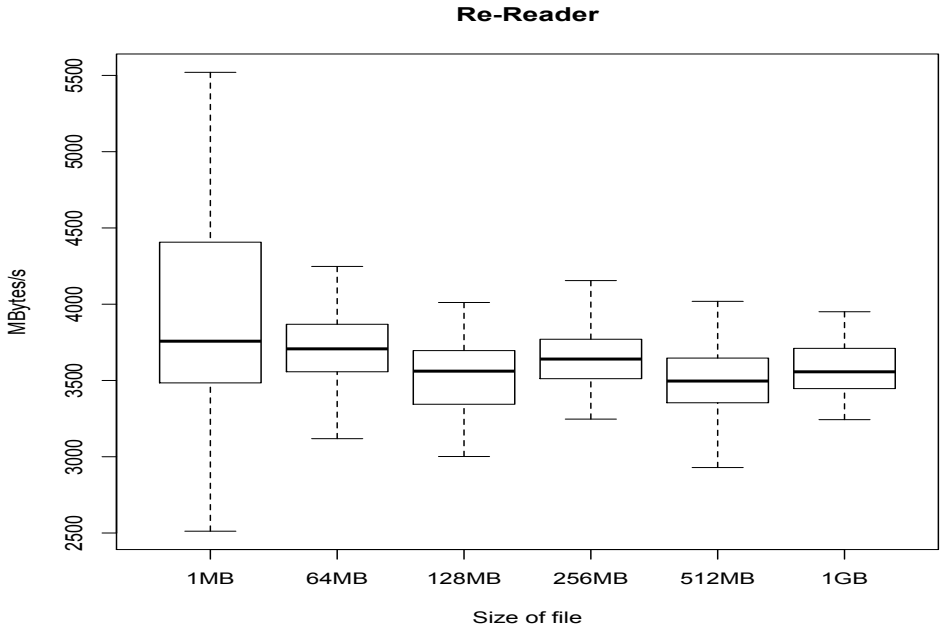

Figure A.17: KVM VM Iozone Re-read

A.2.5 Random Read

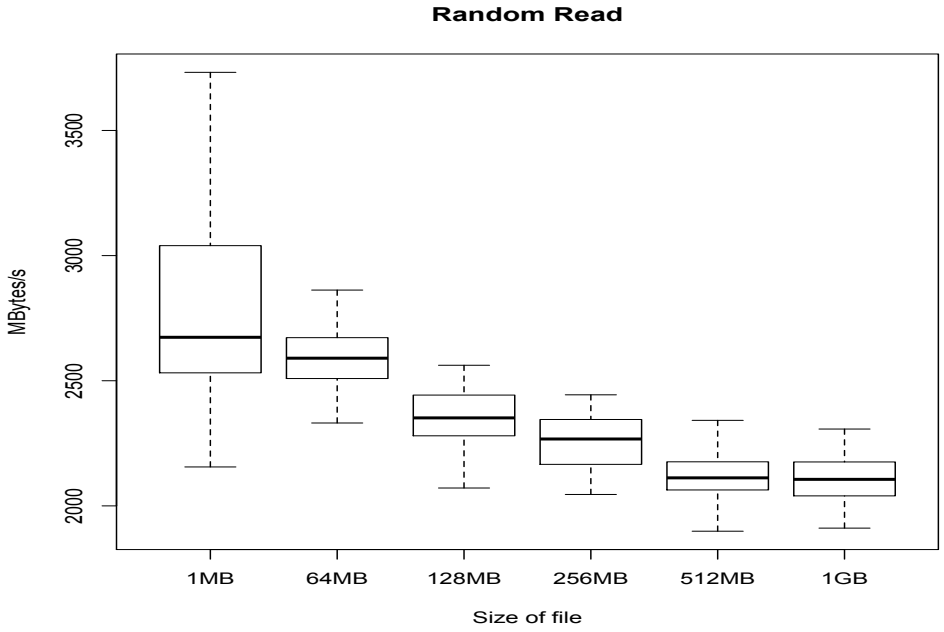

Figure A.18: KVM VM Iozone Random read

A.2.6 Random Write

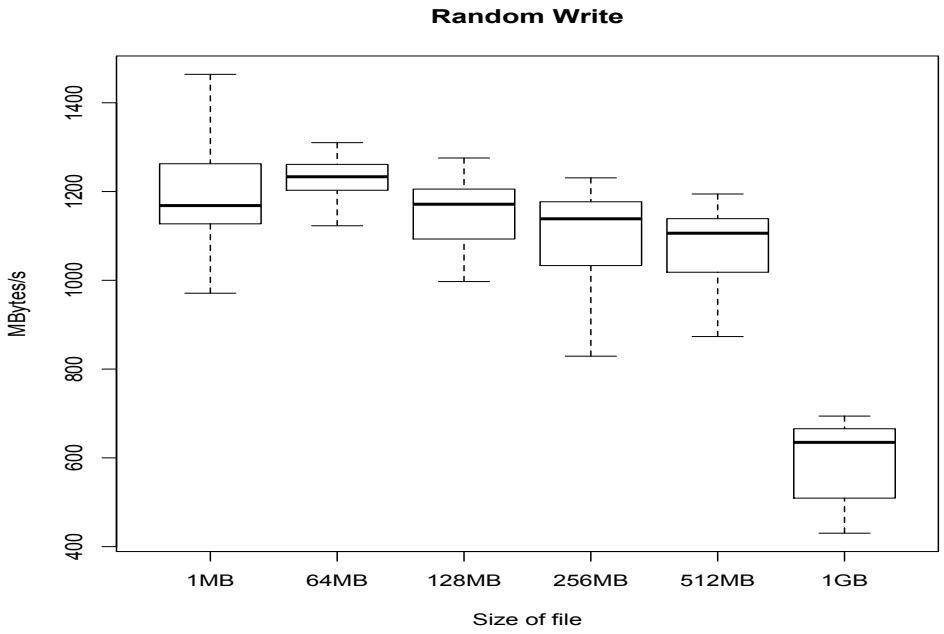

Figure A.19: KVM VM Iozone Random write

A.2.7 Backward Read

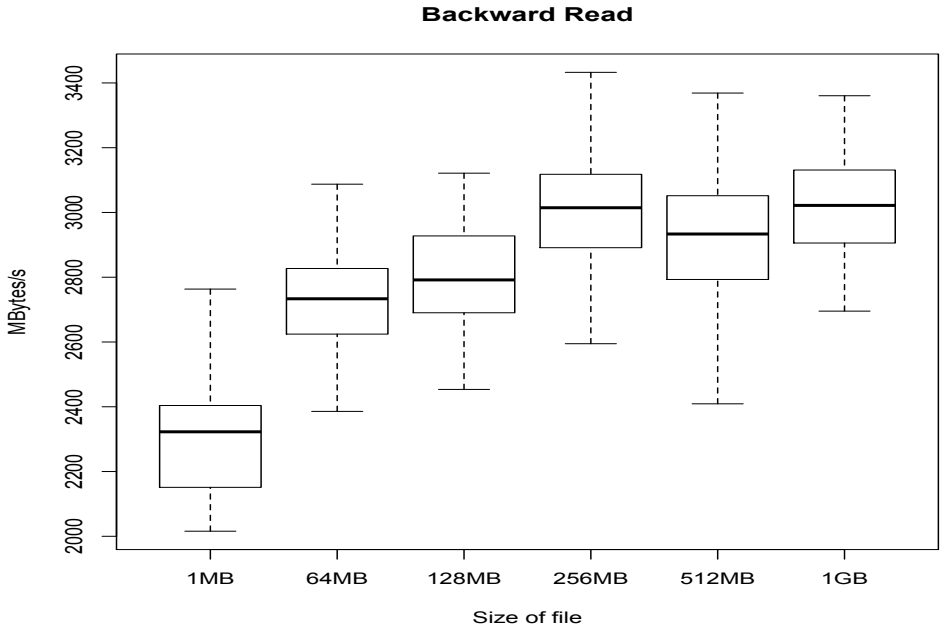

Figure A.20: KVM VM Iozone Backward read

A.2.8 Record Re-write

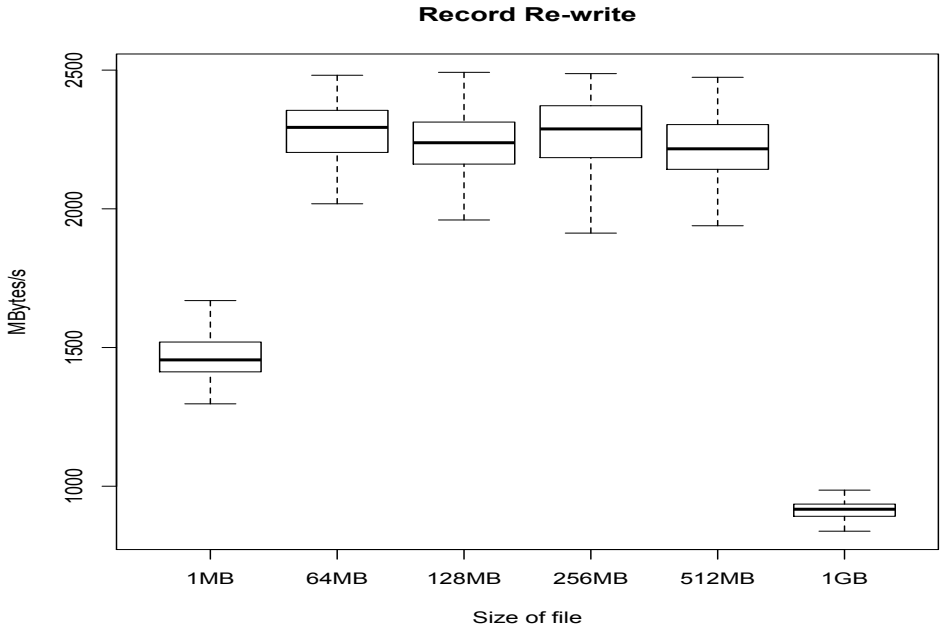

Figure A.21: KVM VM Iozone Record Re-write

A.2.9 Stride Read

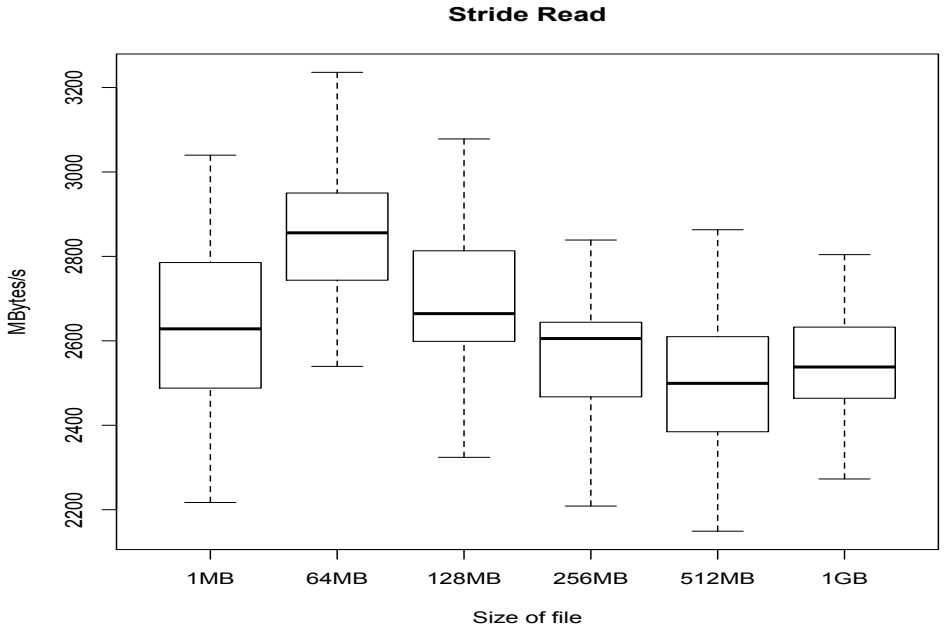

Figure A.22: KVM VM Iozone Stride read

A.2.10 Forward Write

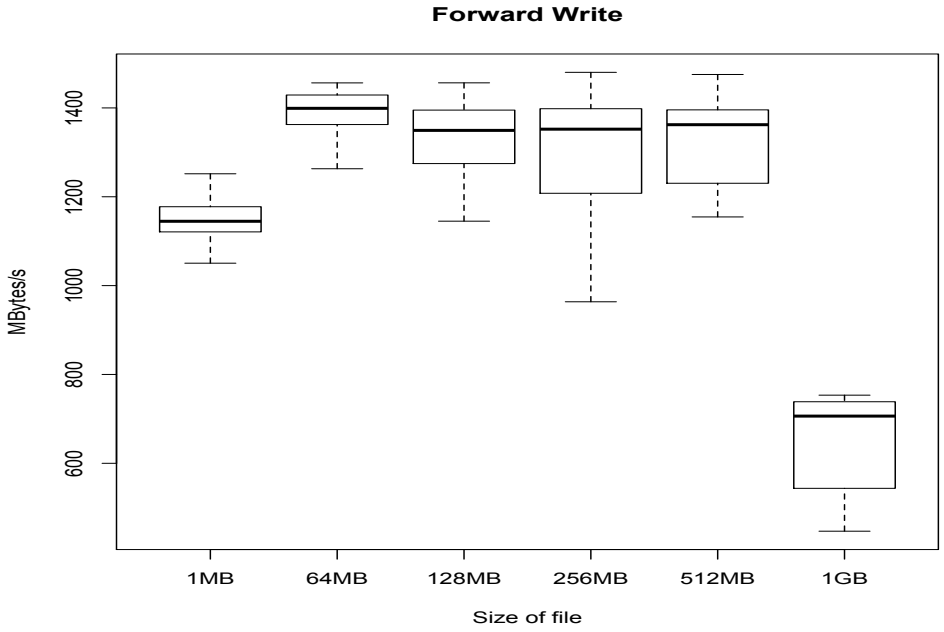

Figure A.23: KVM VM Iozone Forward write

A.2.11 Re-Forward Write

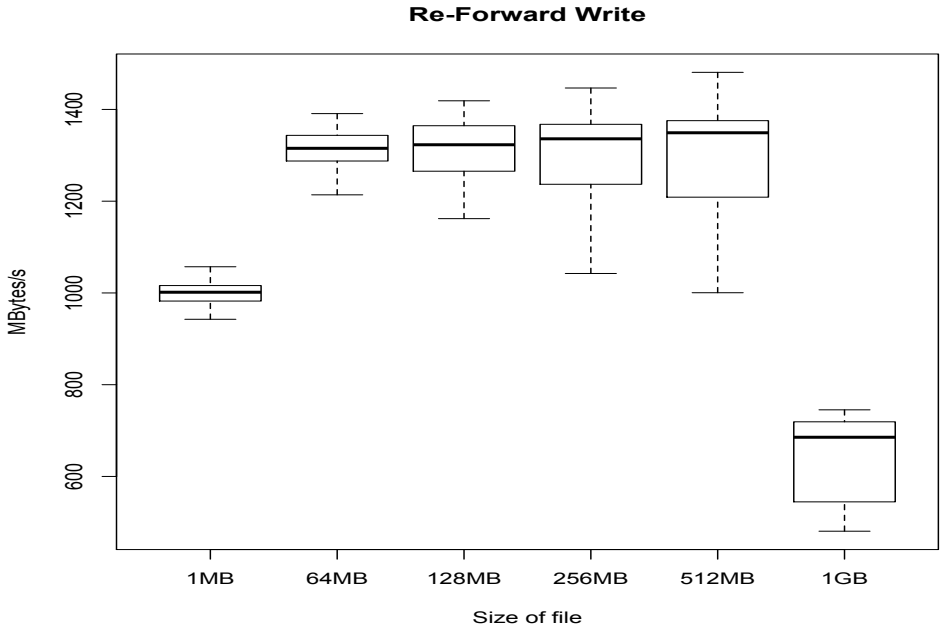

Figure A.24: KVM VM Iozone Re-Forward write

A.2.12 Forward Read

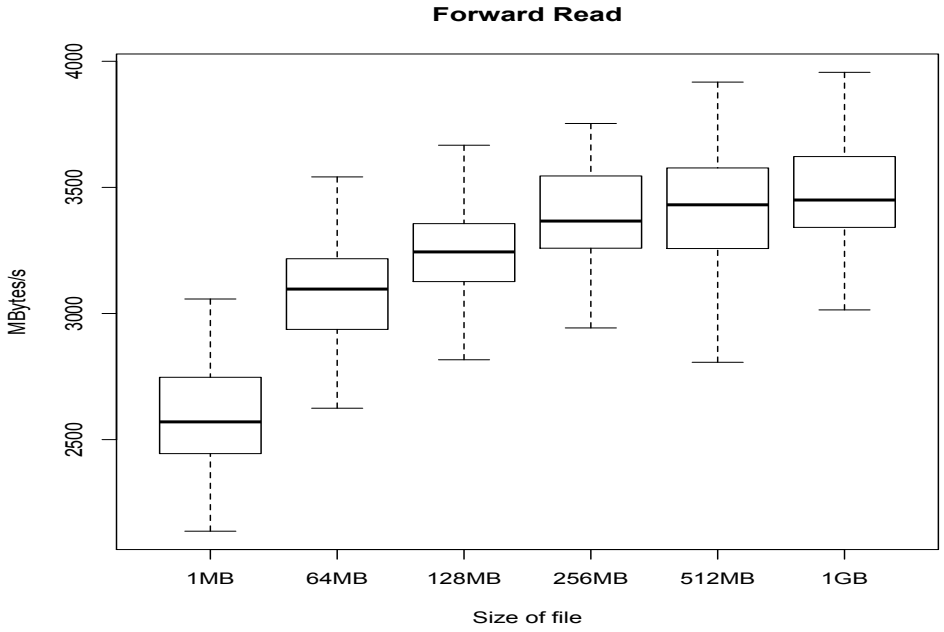

Figure A.25: KVM VM Iozone Forward read

A.2.13 Re-Forward Read

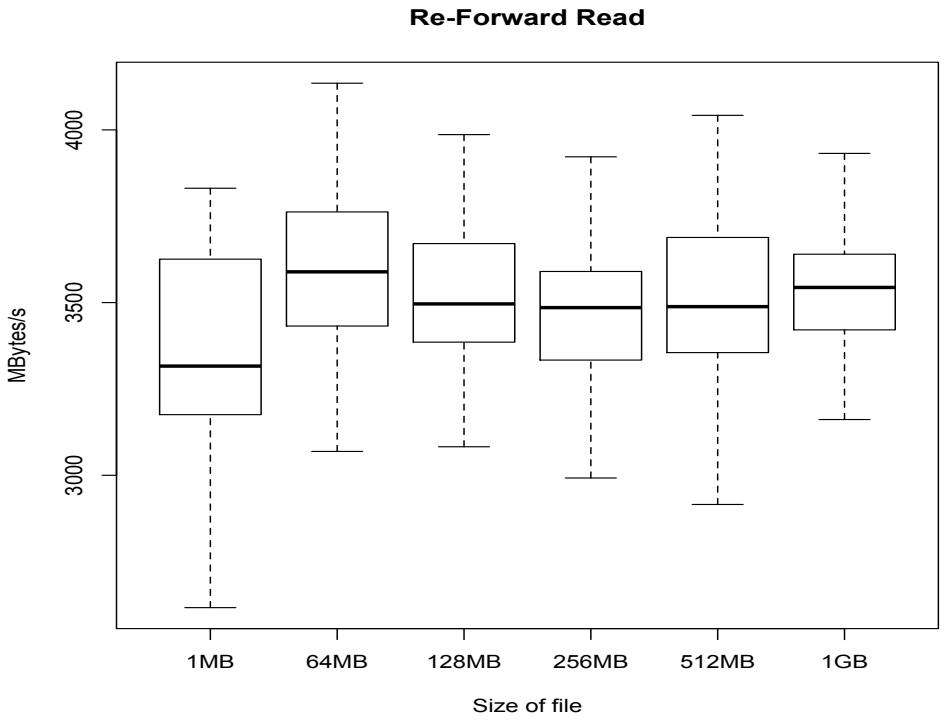

Figure A.26: KVM VM Iozone Re-Forward read

A.3 VMWare Virtual Machine Iozone results

A.3.1 Write

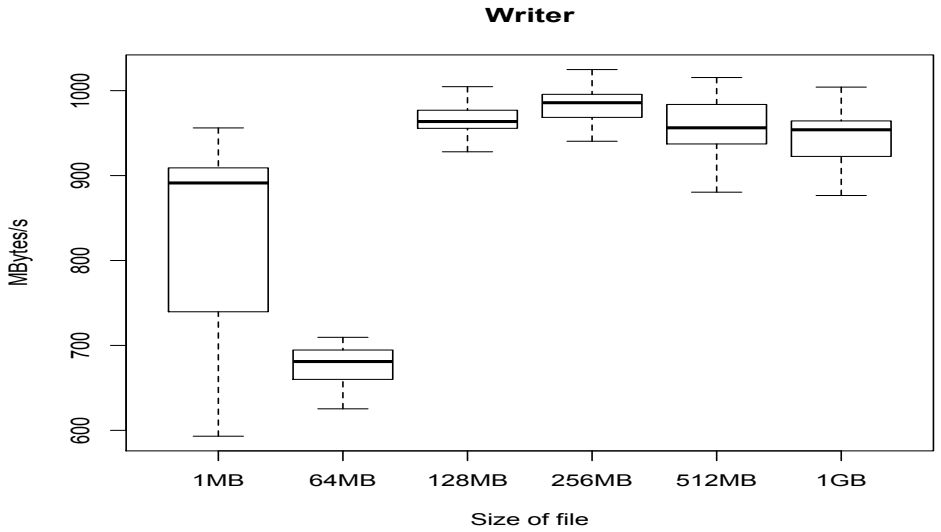

Figure A.27: VMWare VM Iozone Write

A.3.2 Re-Write

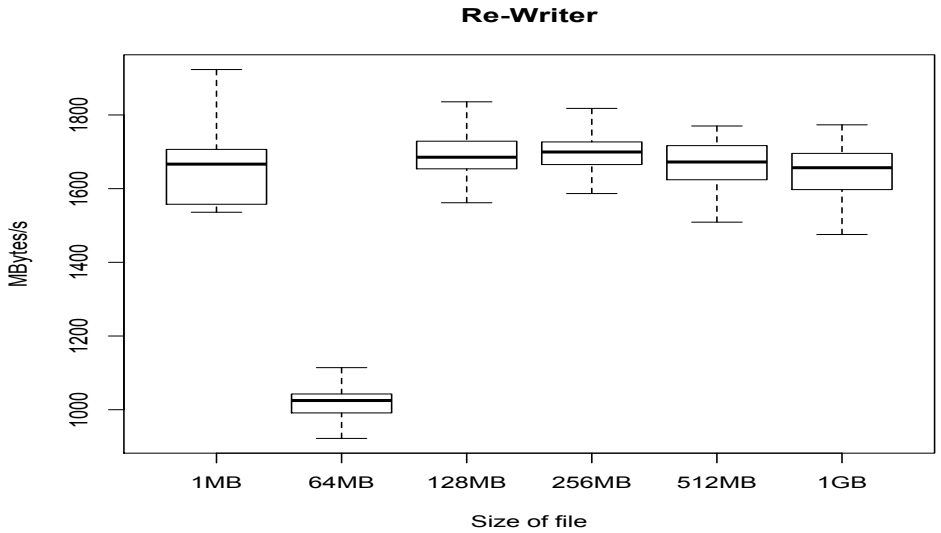

Figure A.28: VMWare VM Iozone Re-write

A.3.3 Read

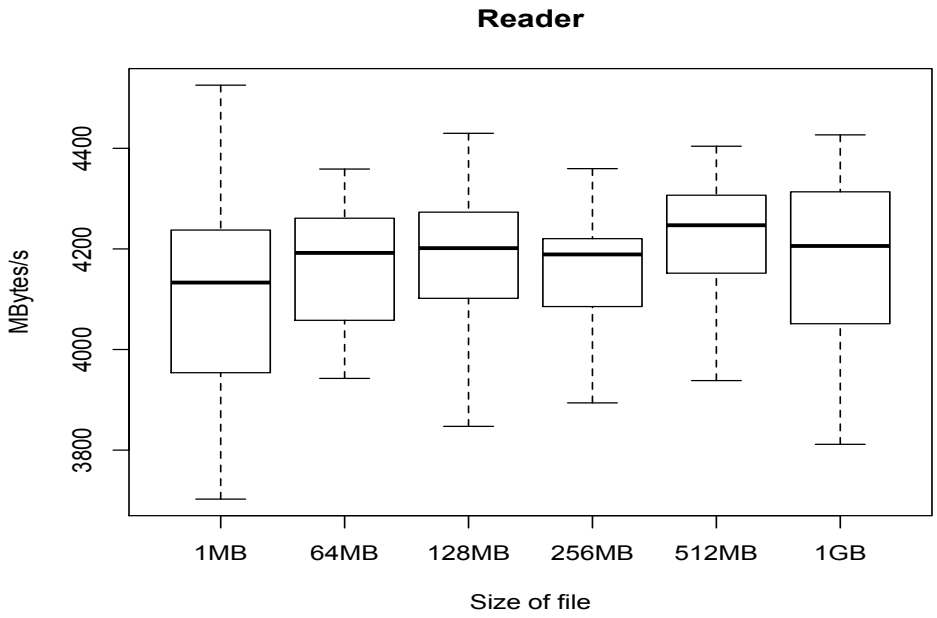

Figure A.29: VMWare VM Iozone Read

A.3.4 Re-Read

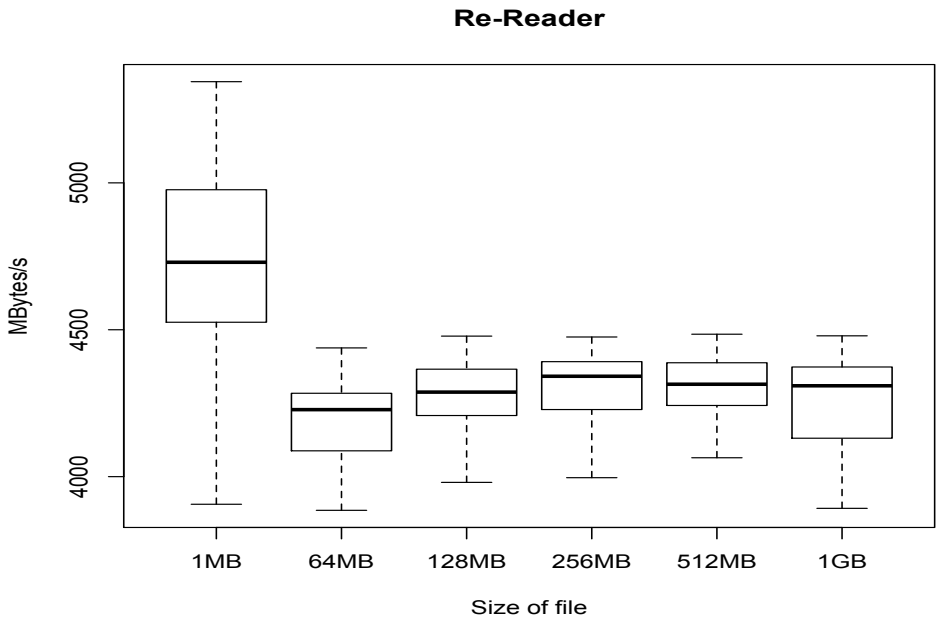

Figure A.30: VMWare VM Iozone Re-read

A.3.5 Random Read

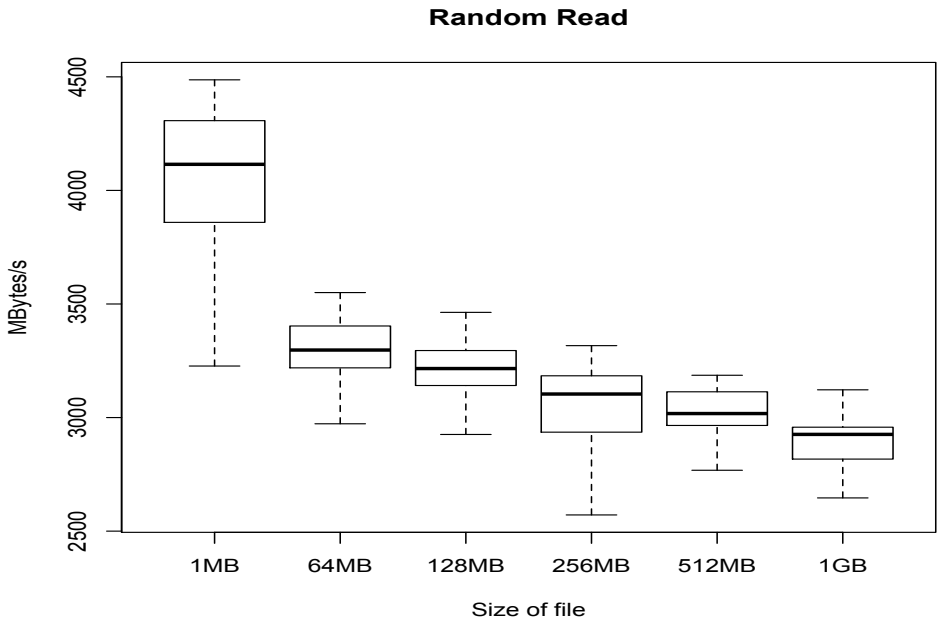

Figure A.31: VMWare VM Iozone Random read

A.3.6 Random Write

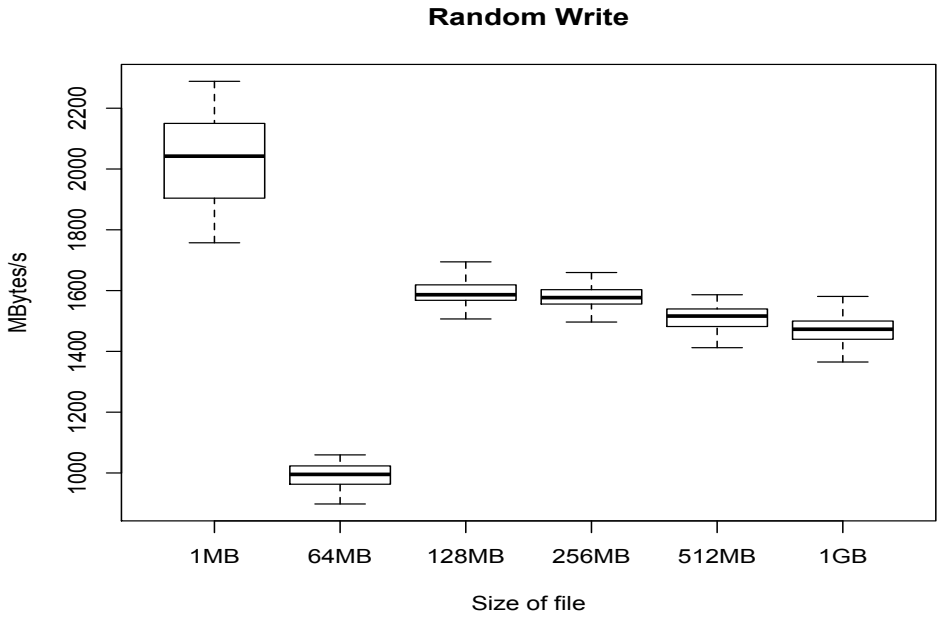

Figure A.32: VMWare VM Iozone Random write

A.3.7 Backward Read

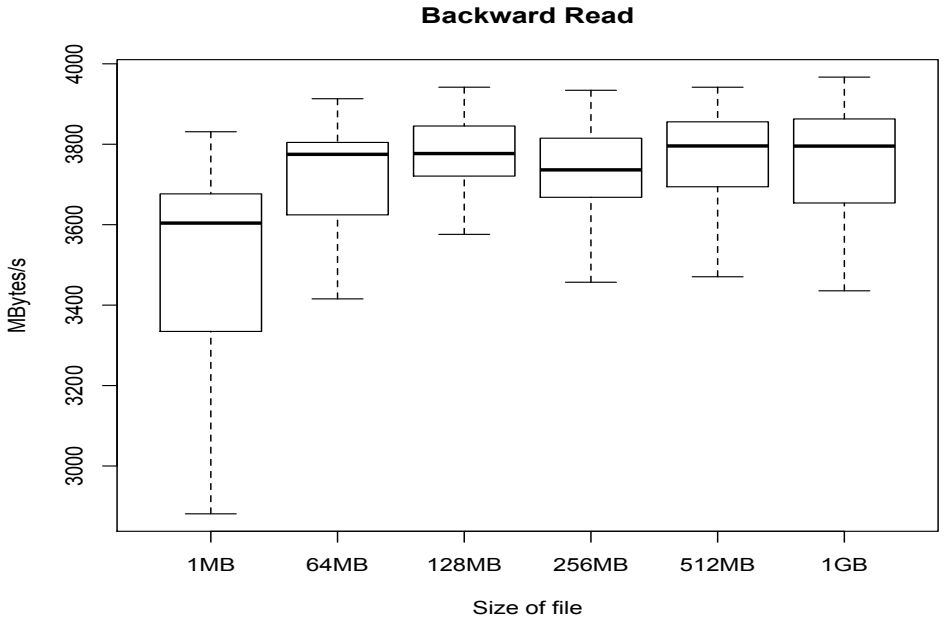

Figure A.33: VMWare VM Iozone Backward read

A.3.8 Record Rewrite

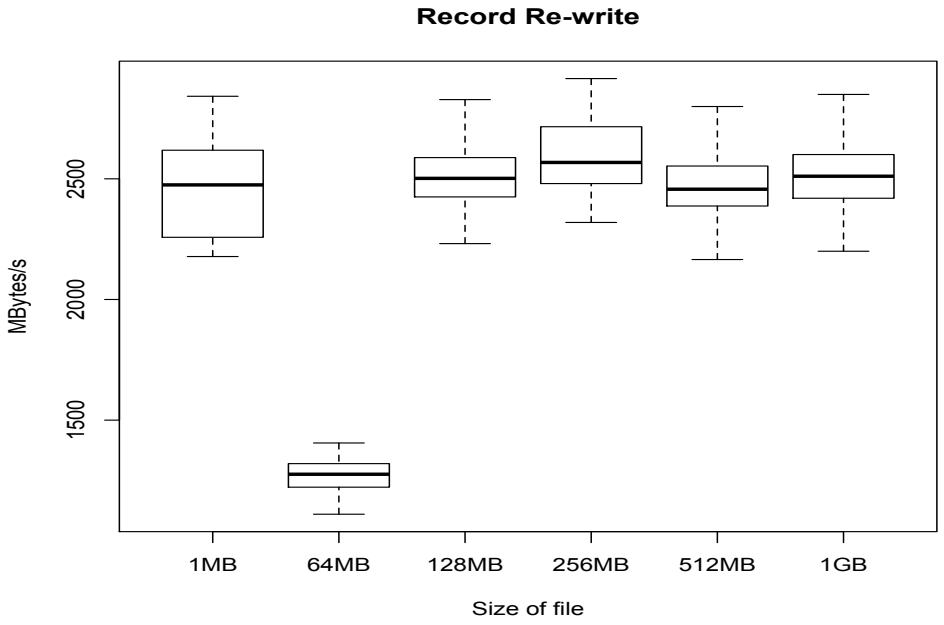

Figure A.34: VMWare VM Iozone Record re-write

A.3.9 Stride Read

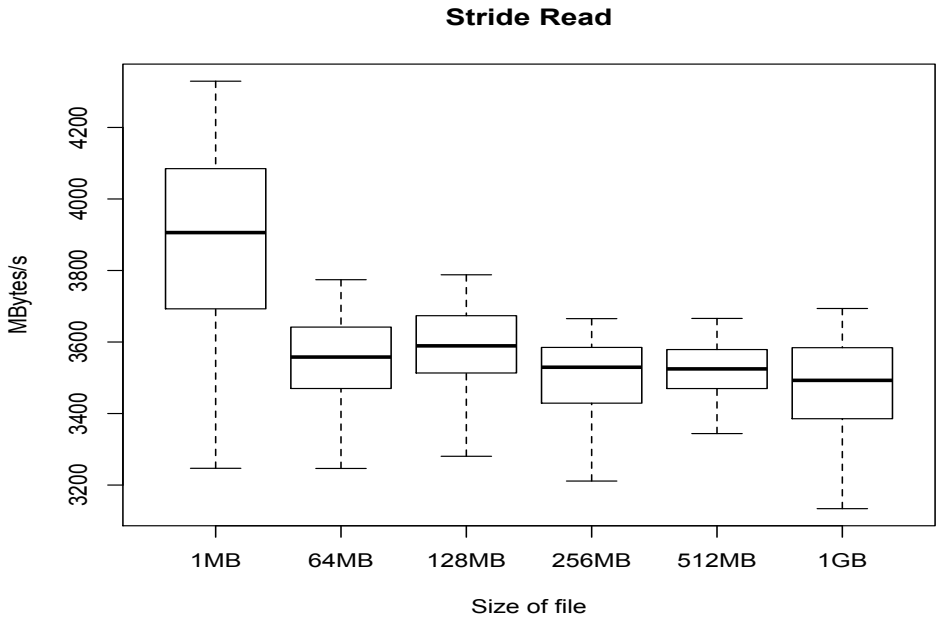

Figure A.35: VMWare VM Iozone Stride read

A.3.10 Forward Write

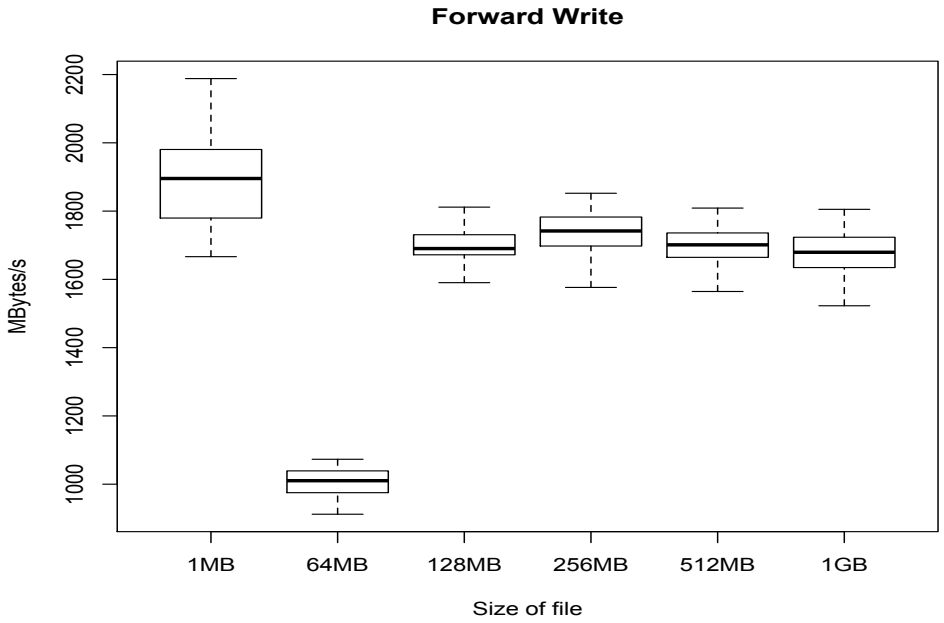

Figure A.36: VMWare VM Iozone Forward write

A.3.11 Re-Forward Write

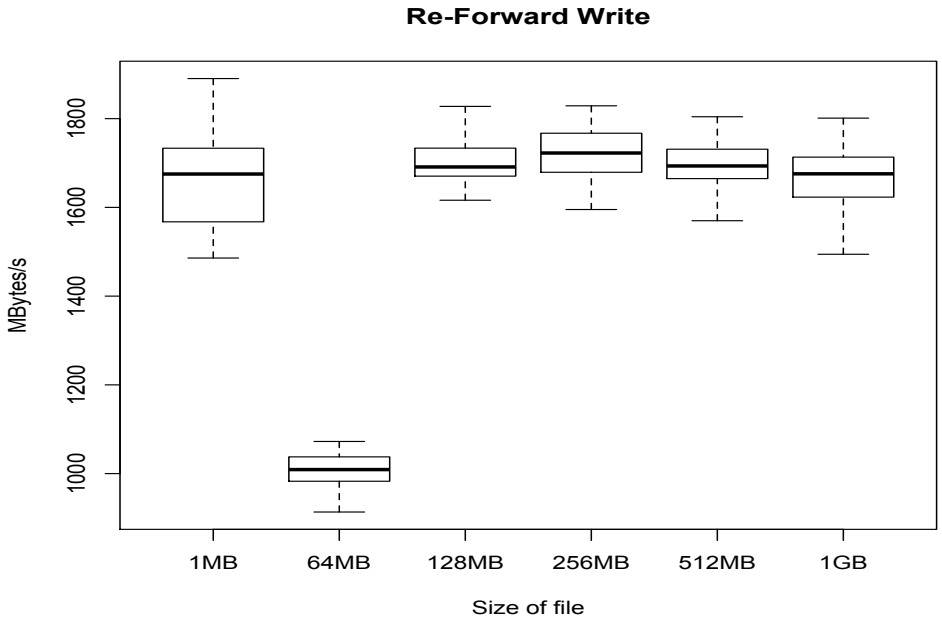

Figure A.37: VMWare VM Iozone Re-Forward write

A.3.12 Forward Read

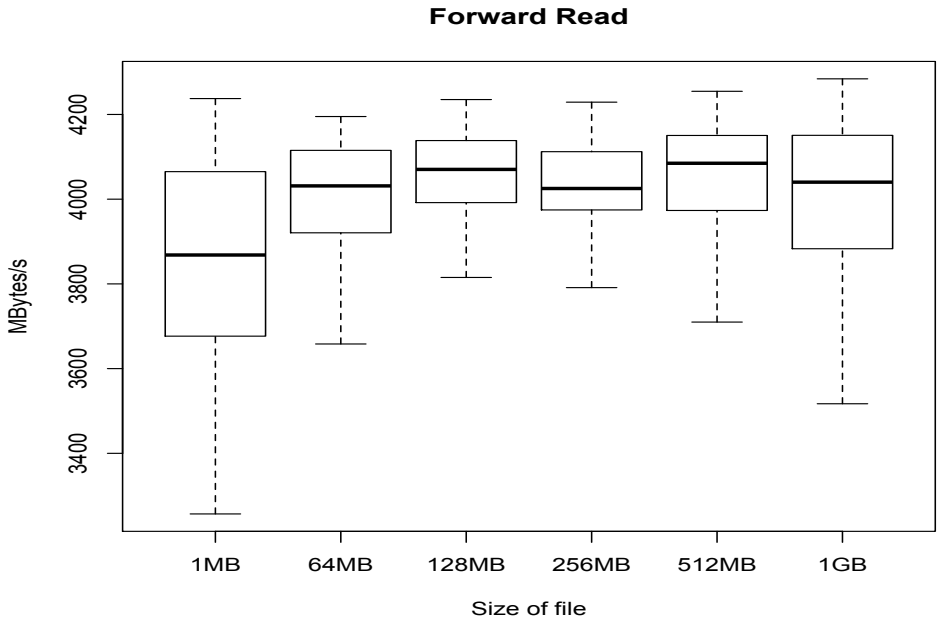

Figure A.38: VMWare VM Iozone Forward read

A.3.13 Re-Forward Read

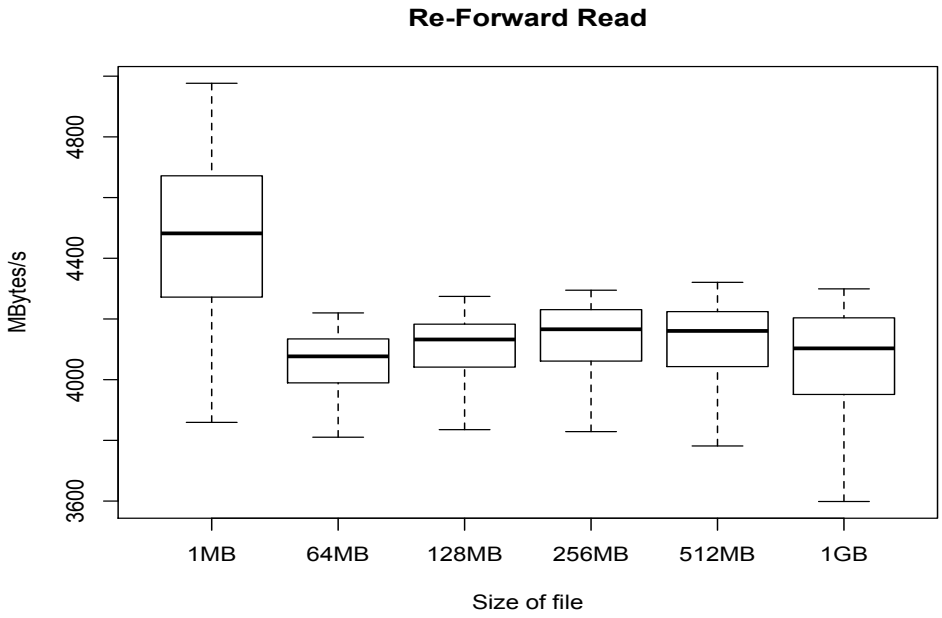

Figure A.39: VMWare VM Iozone Re-Forward read

Appendix B

Ram Speed Results

B.1 Bare Metal Ram speed results

B.1.1 Integer and Writing

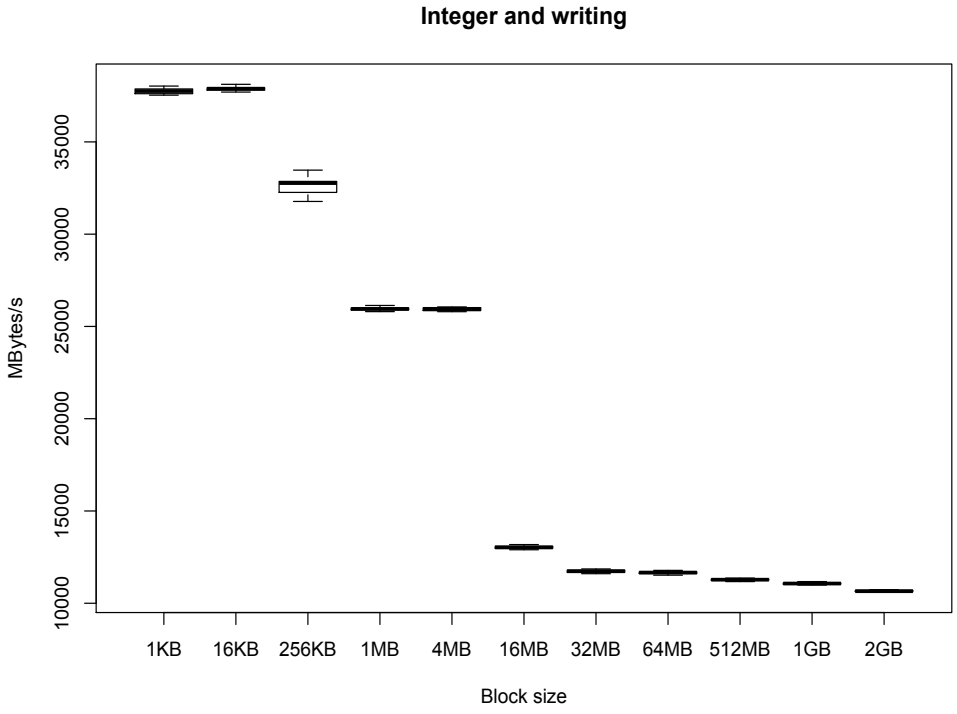

Figure B.1: Bare Metal Ram speed Integer and writing

B.1.2 Integer and Reading

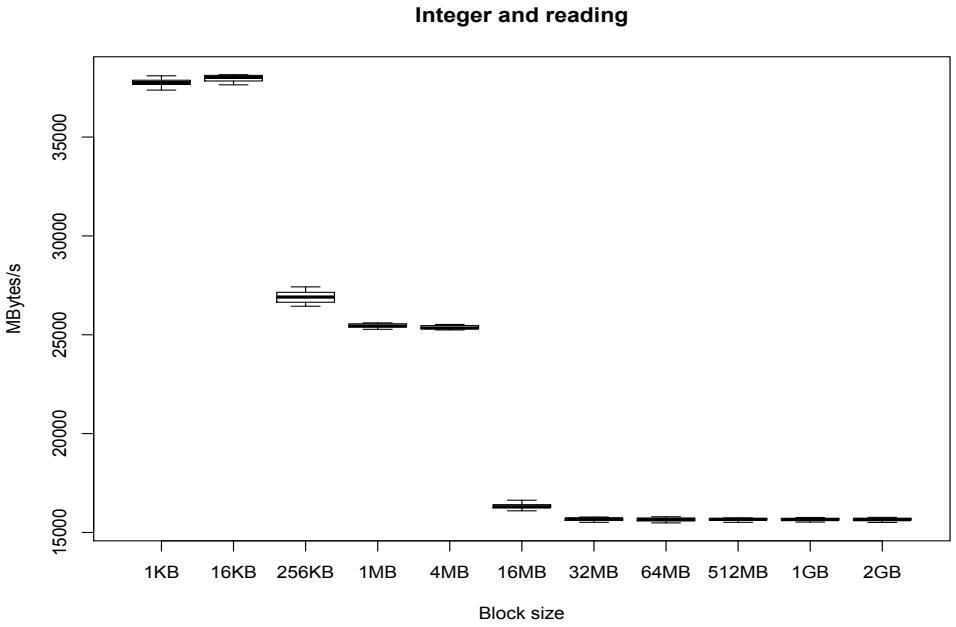

Figure B.2: Bare Metal Ram speed Integer and reading

B.1.3 Float and Writing

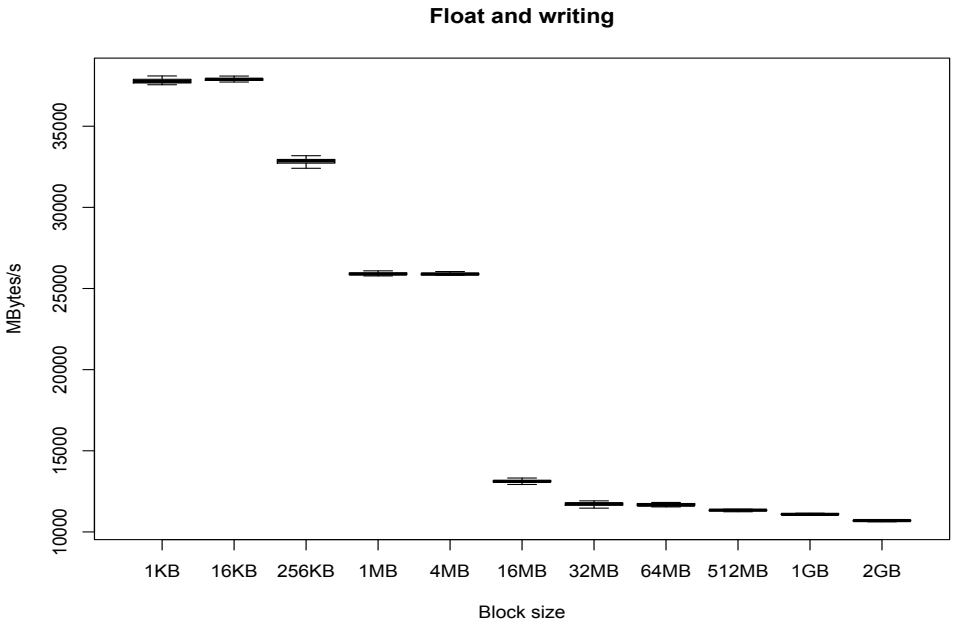

Figure B.3: Bare Metal Ram speed Float and writing

B.1.4 Float and Reading

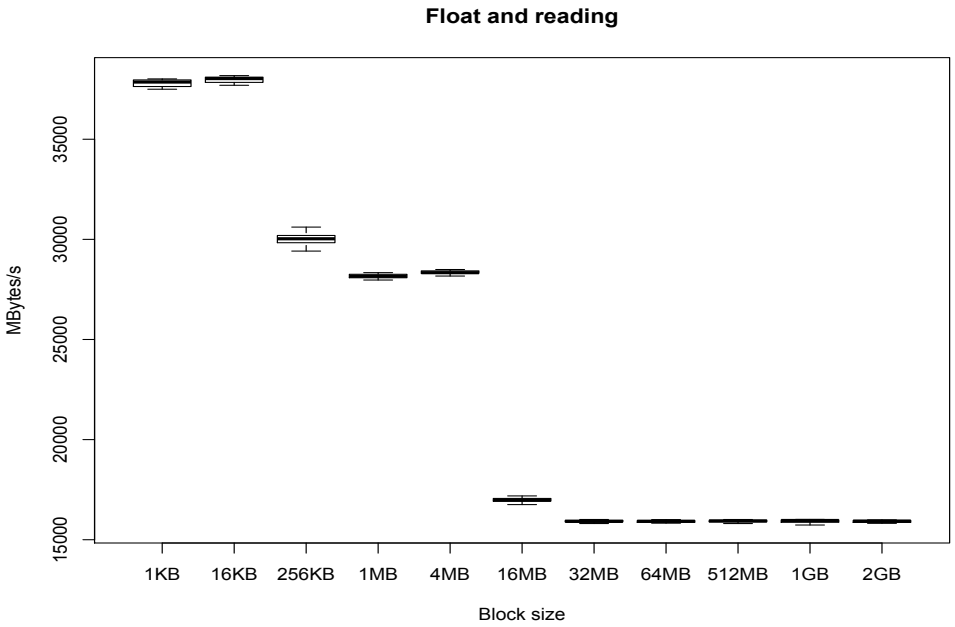

Figure B.4: Bare Metal Ram speed Float and reading

B.2 KVM Virtual Machine Ram speed results

B.2.1 Integer and Writing

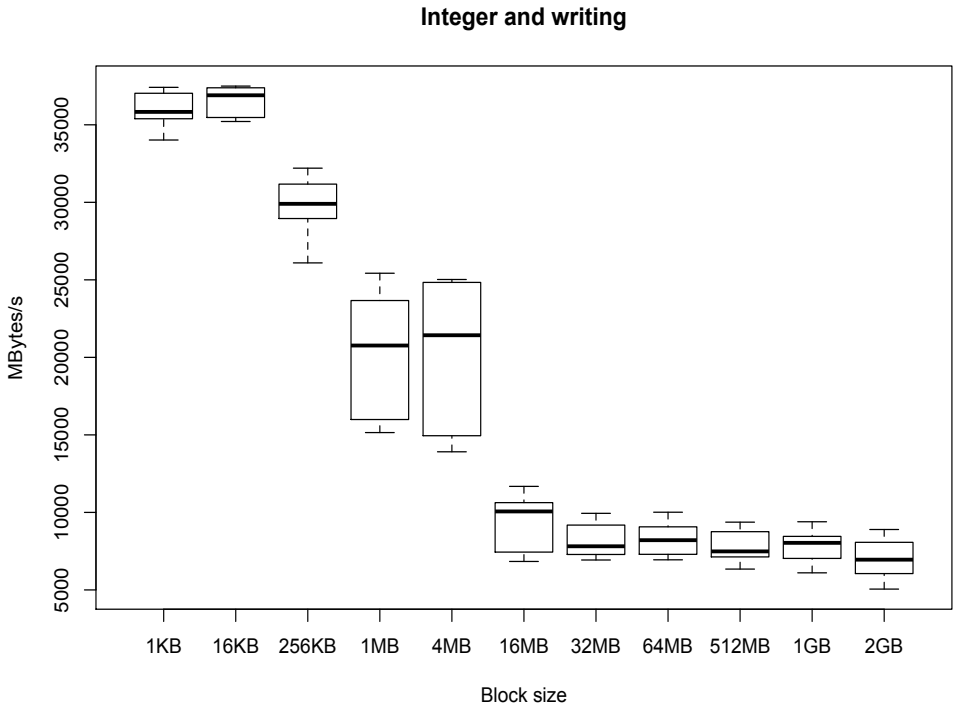

Figure B.5: KVM VM Ram speed Integer and writing

B.2.2 Integer and Reading

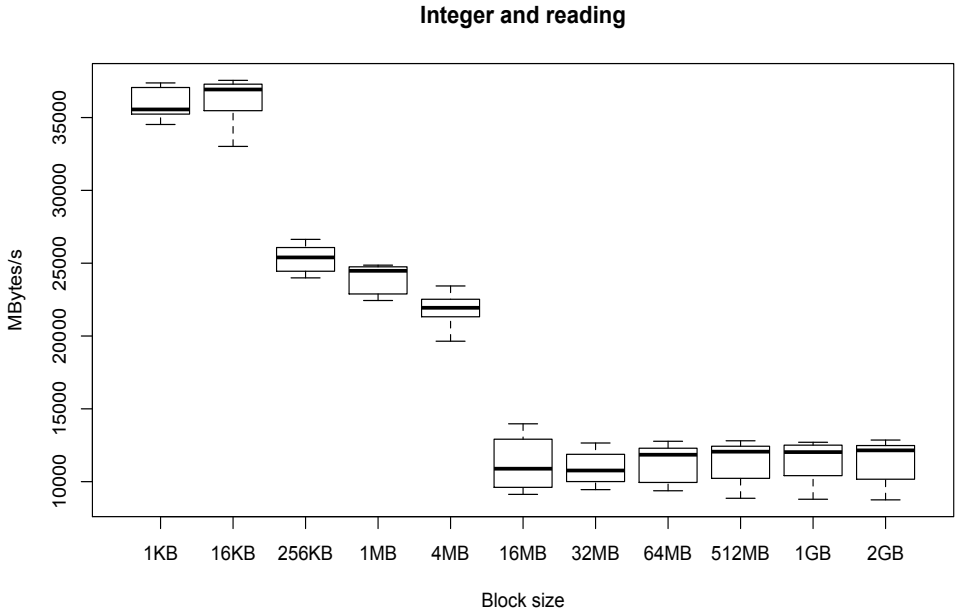

Figure B.6: KVM VM Ram speed Integer and reading

B.2.3 Float and Writing

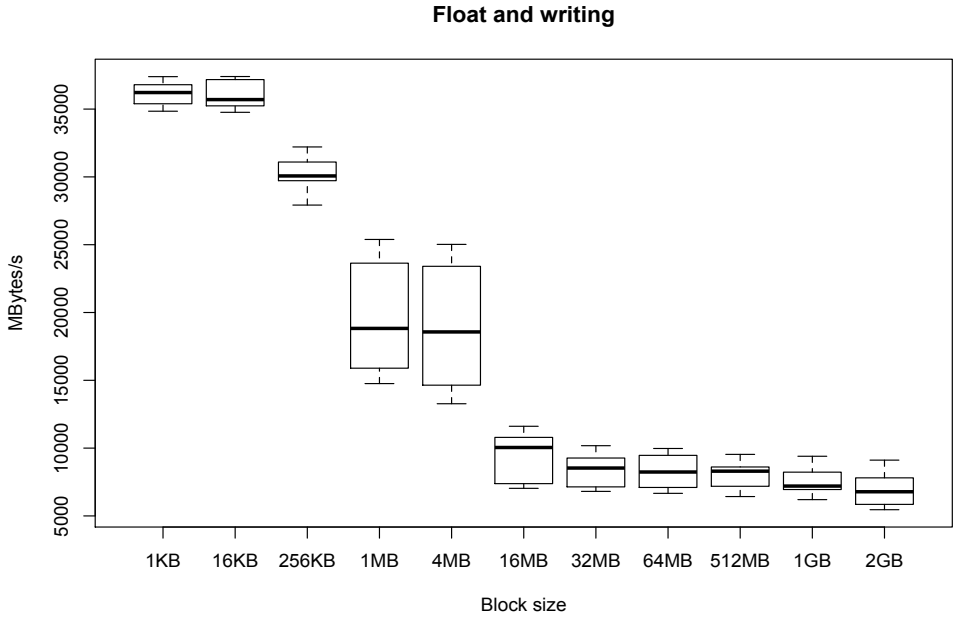

Figure B.7: KVM VM Ram speed Float and writing

B.2.4 Float and Reading

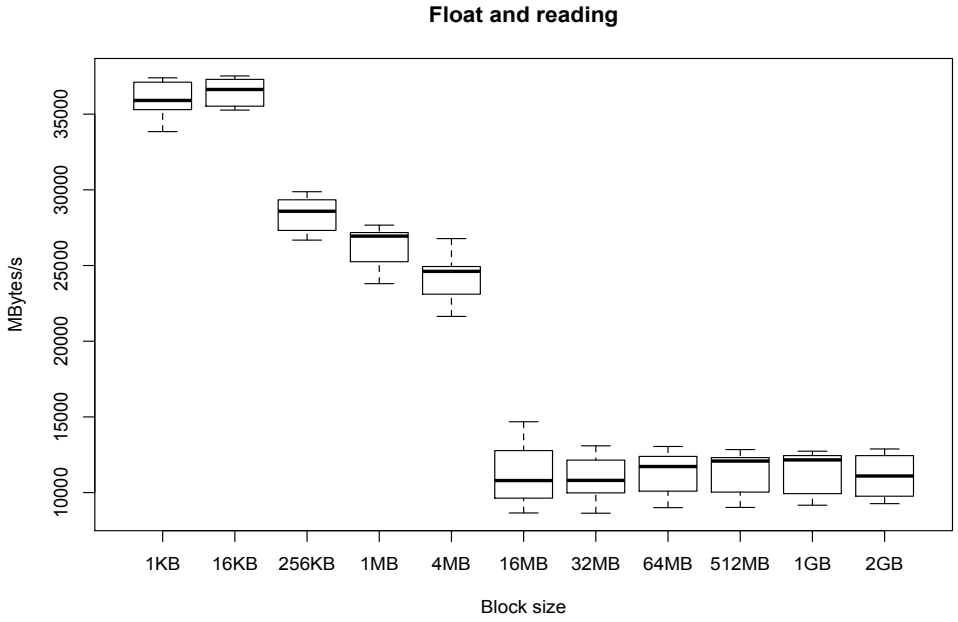

Figure B.8: KVM VM Ram speed Float and reading

B.3 VMWare Virtual Machine Ram speed results

B.3.1 Integer and Writing

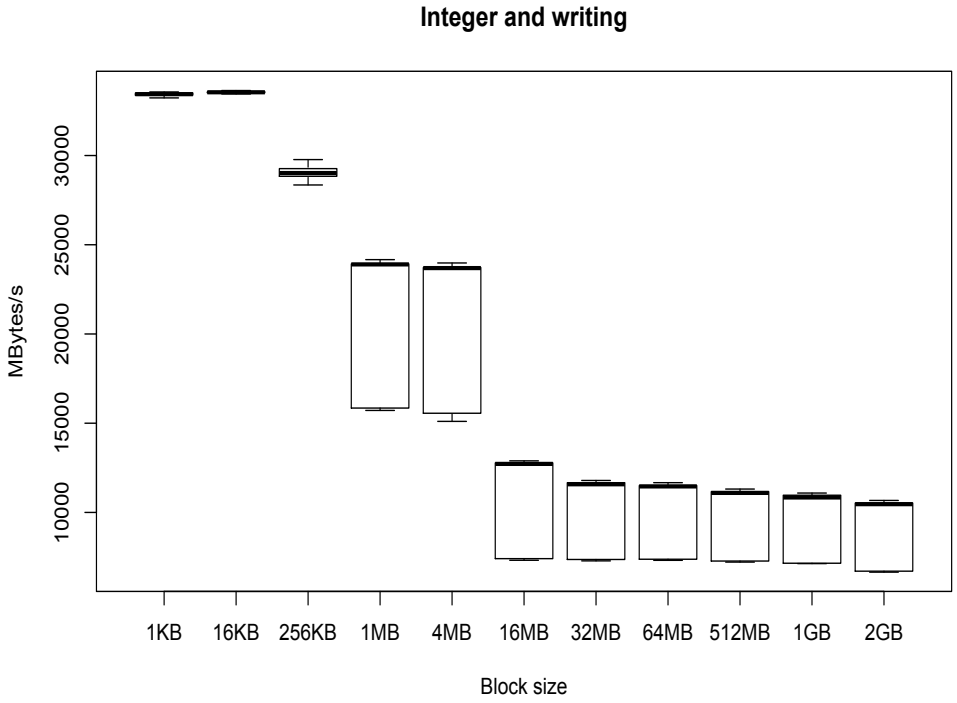

Figure B.9: VMWare VM Ram speed Integer and writing

B.3.2 Integer and Reading

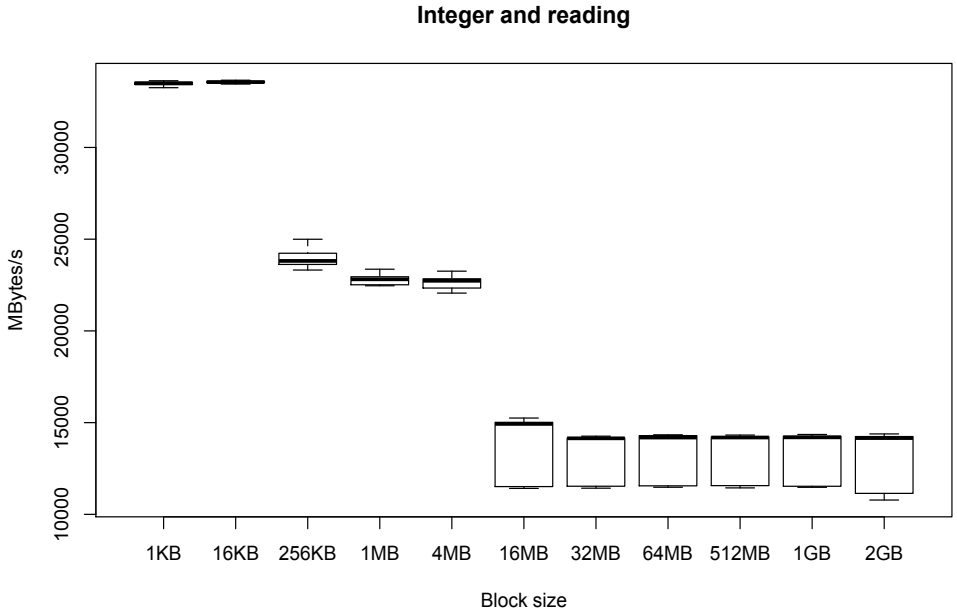

Figure B.10: VMWare VM Ram speed Integer and reading

B.3.3 Float and Writing

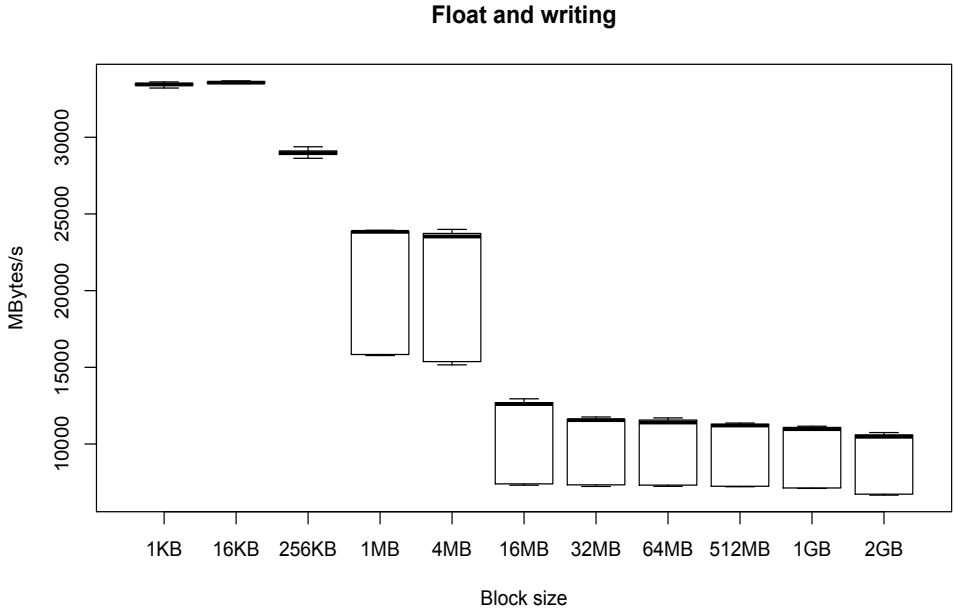

Figure B.11: VMWare VM Ram speed Float and writing

B.3.4 Float and Reading

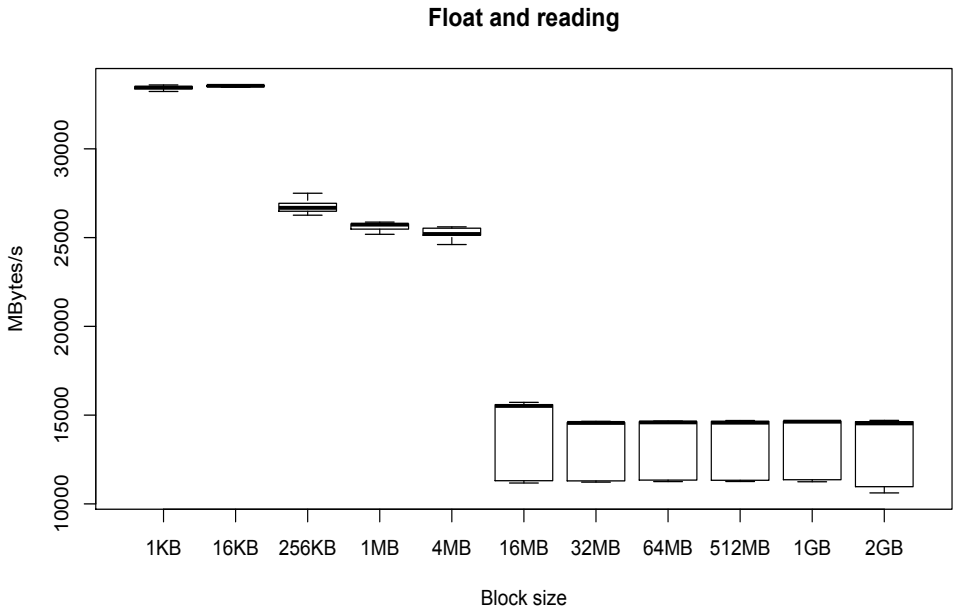

Figure B.12: VMWare VM Ram speed Integer and reading

Appendix C

UnixBench Results

C.1 CPU performance results using UnixBench

Table C.1: CPU performance of UnixBench

System Benchmarks Index Values	Baseline	Results			Index		
		Bare Metal	KVM	VMWare	Bare Metal	KVM	VMWare
Dhrystone 2 using register variables	1.167	1899.769	1782.762	1878.811	0.163	0.153	0.161
Double-Precision Whetstone	0.00055	0.345	0.343	0.338	0.063	0.062	0.061
Execl Throughput	0.00043	0.281	0.153	0.235	0.065	0.036	0.055
File Copy 1024 bufsize 2000 maxblocks	0.0396	4.247	3.152	3.844	0.011	0.008	0.010
File Copy 256 bufsize 500 maxblocks	0.01655	1.155	0.855	1.036	0.007	0.005	0.006
File Copy 4096 bufsize 8000 maxblocks	0.058	13.100	9.746	12.262	0.023	0.017	0.021
Pipe Throughput	0.1244	94.015	61.230	106.423	0.076	0.017	0.086
Pipe-based Context Switching	0.040	24.793	12.525	20.622	0.062	0.031	0.052
Process Creation	0.00126	0.762	0.472	0.581	0.060	0.037	0.046
Shell Scripts (1 concurrent)	0.000424	0.395	0.272	0.351	0.093	0.064	0.083
Shell Scripts (8 concurrent)	0.00006	0.054	0.038	0.049	0.090	0.064	0.082
System Call Overhead	0.15000	69.229	68.337	68.682	0.046	0.046	0.046
System Benchmarks Index Score					0.047	0.034	0.043

C.2 CPU performance results using UnixBench for each run

Table C.2: CPU performance of UnixBench each run

System Benchmarks Index Values	Bare Metal			KVM			VMWare		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
Dhrystone 2 using register variables	189940.5	190027.3	189962.9	174454.9	174132.8	186240.9	187707.0	187935.1	188001.2
Double-Precision Whetstone	34.5	34.5	34.4	34.4	34.4	34.2	33.9	33.8	33.8
Execl Throughput	28.0	28.3	28.1	15.1	15.3	15.6	23.5	23.5	23.5
File Copy 1024 bufsize 2000	429.6	425.6	418.8	324.5	307.6	313.3	385.8	386.2	381.4
File Copy 256 bufsize 500	117.2	117.9	111.4	86.6	82.2	87.8	103.3	104.2	103.4
File Copy 4096 bufsize 8000	1300.7	1317.7	1311.6	978.5	936.1	1009.1	1226.5	1226.7	1225.4
Pipe Throughput	9435.1	9460.2	9309.4	6397.3	6147.2	5824.4	10578.0	10354.7	10994.1
Pipe-based Context Switching	2487.1	2492.4	2458.5	1269.6	1257.9	1230.0	2079.6	2003.8	2103.3
Process Creation	76.3	76.5	75.8	47.1	48.0	46.4	58.2	57.7	58.2
Shell Scripts (1 concurrent)	39.6	39.6	39.5	27.2	26.5	28.1	35.2	35.0	35.2
Shell Scripts (8 concurrent)	5.4	5.4	5.4	3.8	3.8	3.9	4.9	4.9	4.9
System Call Overhead	6657.0	7380.3	6731.4	6921.0	6824.5	6755.6	6783.0	6842.6	6979.2

Appendix D

Script for Iozone

D.1 Appendix: Iozonetest.sh

The Iozonetest.sh script

```
#! /bin/bash
COUNTER=0
while [ $COUNTER -lt 70 ]; do
        iozone -s 1024M -r 4k -Rac >> iozonetest.txt
        let COUNTER=COUNTER+1
        sleep 5
        echo $COUNTER
done
```

D.2 Appendix: Fileseprator.sh

The Fileseprator.sh script

```
#! /usr/bin/perl
$teller = 0;
open (FIL, "./iozonetest.txt") or
die "cant open file /iozonetest.txt\n";

open (WRITER, ">>iozoneWriter.txt") or die \\
"cant open file iozoneWriter.txt\n";

open (WRITER1, ">>iozoneWriterExcel.txt") or die \\
"cant open file iozoneWriterExcel.txt\n";

open (REWRITER, ">>iozoneRe-writer.txt") or die \\
"cant open file iozoneRewriter.txt\n";

open (REWRITER1, ">>iozoneRe-writerExcel.txt") or die \\
"cant open file iozoneRe-writerExcel.txt\n";

open (READER, ">>iozoneReader.txt") or die \\
"cant open file iozoneReader.txt\n";

open (READER1, ">>iozoneReaderExcel.txt") or die \\
"cant open file iozoneReaderExcel.txt\n";

open (REREADER, ">>iozoneRe-Reader.txt") or die \\
"cant open file iozoneRe-Reader.txt\n";

open (REREADER1, ">>iozoneRe-ReaderExcel.txt") or die \\
"cant open file iozoneRe-ReaderExcel.txt\n";

```

```
30  open (RANDOMREAD, ">>iozoneRandomRead.txt") or die \\
31  "cant open file iozoneRandomRead.txt\n";
32
33  open (RANDOMREAD1, ">>iozoneRandomReadExcel.txt") or die \\
34  "cant open file iozoneRandomReadExcel.txt\n";
35
36  open (RANDOMWRITE, ">>iozoneRandomwrite.txt") or die \\
37  "cant open file iozoneRandomwrite.txt\n";
38
39  open (RANDOMWRITE1, ">>iozoneRandomwriteExcel.txt") or die \\
40  "cant open file iozoneRandomwriteExcel.txt\n";
41
42  open (BACKWARDREAD, ">>iozoneBackwardread.txt") or die \\
43  "cant open file iozoneBackwardread.txt\n";
44
45  open (BACKWARDREAD1, ">>iozoneBackwardreadExcel.txt") or die \\
46  "cant open file iozoneBackwardreadExcel.txt\n";
47
48  open (RECORDREWRITE, ">>iozoneRecordrewrite.txt") or die \\
49  "cant open file iozoneRecordrewrite.txt\n";
50
51  open (RECORDREWRITE1, ">>iozoneRecordrewriteExcel.txt") or die \\
52  "cant open file iozoneRecordrewriteExcel.txt\n";
53
54  open (STRIDEREAD, ">>iozoneStridread.txt") or die \\
55  "cant open file iozoneStridread.txt\n";
56
57  open (STRIDEREAD1, ">>iozoneStridreadExcel.txt") or die \\
58  "cant open file iozoneStridreadExcel.txt\n";
59
60  open (FWRITE, ">>iozoneFwrite.txt") or die \\
61  "cant open file iozoneFwrite.txt\n";
62
63  open (FWRITE1, ">>iozoneFwriteExcel.txt") or die \\
64  "cant open file iozoneFwriteExcel.txt\n";
65
66  open (REFWRITE, ">>iozoneRe-Fwrite.txt") or die \\
67  "cant open file iozoneRe-Fwrite.txt\n";
68
69  open (REFWRITE1, ">>iozoneRe-FwriteExcel.txt") or die \\
70  "cant open file iozoneRe-FwriteExcel.txt\n";
71
72  open (FREAD, ">>iozoneFread.txt") or die \\
73  "cant open file iozoneFread.txt\n";
74
75  open (FREAD1, ">>iozoneFreadExcel.txt") or die \\
76  "cant open file iozoneFreadExcel.txt\n";
77
78  open (REFREAD, ">>iozoneRe-Fread.txt") or die \\
79  "cant open file iozoneRe-Fread.txt\n";
80
81  open (REFREAD1, ">>iozoneRe-FreadExcel.txt") or die \\
82  "cant open file iozoneRe-FreadExcel.txt\n";
83
84  while($line = <FIL>)
85  {
86
87   if($line =~ /^"Writer report"/)
88   {
89    $line1 = <FIL>; #reads a line of no interrest
90    #line containing the files size and Mbps value
91    $line1 = <FIL>;
92    @array = split(" ",$line1); #splits the line
93    #writes the Mbps value to the file
94    print WRITER1 "$array[1]\n";
95    print WRITER "{$teller,";
```

```
 print WRITER "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Re-writer report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print REWRITER1 "$array[1]\n";
 print REWRITER "{$teller,";
 print REWRITER "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Reader report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print READER1 $array[1];
 print READER1 "\n";
 print READER "{$teller,";
 print READER "$array[1]},";
 $teller +=1;
}

if($line =~ /^"Re-Reader report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print REREADER1 "$array[1]\n";
 print REREADER "{$teller,";
 print REREADER "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Random read report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print RANDOMREAD1 "$array[1]\n";
 print RANDOMREAD "{$teller,";
 print RANDOMREAD "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Random write report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print RANDOMWRITE1 "$array[1]\n";
 print RANDOMWRITE "{$teller,";
 print RANDOMWRITE "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Backward read report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print BACKWARDREAD1 "$array[1]\n";
```

```
 print BACKWARDREAD "{$teller,";
 print BACKWARDREAD "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Record rewrite report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print RECORDREWRITE1 "$array[1]\n";
 print RECORDREWRITE "{$teller,";
 print RECORDREWRITE "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Stride read report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print STRIDEREAD1 "$array[1]\n";
 print STRIDEREAD "{$teller,";
 print STRIDEREAD "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Fwrite report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print FWRITE1 "$array[1]\n";
 print FWRITE "{$teller,";
 print FWRITE "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Re-Fwrite report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print REFWRITE1 "$array[1]\n";
 print REFWRITE "{$teller,";
 print REFWRITE "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Fread report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print FREAD1 "$array[1]\n";
 print FREAD "{$teller,";
 print FREAD "$array[1]},";
 $teller += 1;
}

if($line =~ /^"Re-Fread report"/)
{
 $line1 = <FIL>;
 $line1 = <FIL>;
 @array = split(" ",$line1);
 print REFREAD1 "$array[1]\n";
```

```
  print REFREAD "{$teller,";
  print REFREAD "$array[1]},";
  $teller += 1;
 }
}
close FIL;
close WRITER;
close WRITER1;
close REWRITER;
close REWRITER1;
close READER;
close READER1;
close REREADER;
close REREADER1;
close RANDOMREAD;
close RANDOMREAD1;
close RANDOMWRITE;
close RANDOMWRITE1;
close BACKWARDREAD;
close BACKWARDREAD1;
close RECORDREWRITE;
close RECORDREWRITE1;
close STRIDEREAD;
close STRIDEREAD1;
close FWRITE;
close FWRITE1;
close REFWRITE;
close REFWRITE1;
close FREAD;
close FREAD1;
close REFREAD;
close REFREAD1;
```

D.3 Appendix: R commands for Iozone

The R commands for iozone

```
> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/1 MB/
iozoneRe-writerExcel.txt")
> Rewriter1mb = values$V1
> Rewriter1mb = Rewriter1mb/1024
> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/64 MB/
iozoneRe-writerExcel.txt")
> Rewriter64mb = values$V1
> Rewriter64mb = Rewriter64mb/1024
> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/128 MB/
iozoneRe-writerExcel.txt")
> Rewriter128mb = values$V1
> Rewriter128mb = Rewriter128mb/1024

> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/256 MB/
iozoneRe-writerExcel.txt")
> Rewriter256mb = values$V1
> Rewriter256mb = Rewriter256mb/1024
> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/512 MB/
iozoneRe-writerExcel.txt")
> Rewriter512mb = values$V1
> Rewriter512mb = Rewriter512mb/1024
> values = read.table("/Users/naveedy/Desktop/IOZone RHEL/iozone with all optns/1 GB/
iozoneRe-writerExcel.txt")
> Rewriter1gb = values$V1
> Rewriter1gb = Rewriter1gb/1024
> summary(Rewriter1mb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
   1091    1374    1430    1582    1818    1930
> summary(Rewriter64mb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1455    1642    1679    1674    1715    1787
> summary(Rewriter128mb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1481    1669    1704    1698    1737    1788
> summary(Rewriter256mb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1707    1844    1865    1859    1876    1907
> summary(Rewriter512mb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1723    1854    1866    1867    1884    1912
> summary(Rewriter1gb)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1835    1871    1881    1881    1892    1921
> boxplot(Rewriter1mb,Rewriter64mb,Rewriter128mb,Rewriter256mb,Rewriter512mb,Rewriter1gb,
 outline=FALSE, ylab="MBytes/s", xlab="Size of file", names=c("1MB","64MB,"128MB,"256MB",
"512MB","1GB"), main="Re writer")
> jpeg("/Users/naveedy/Rewriter.jpeg")
> boxplot(Rewriter1mb,Rewriter256mb,Rewriter512mb,Rewriter1gb, outline=FALSE, ylab="MBytes/s",
 xlab="Size of file", names=c("1MB","64MB",128MB",256MB","512MB","1GB"), main="Re writer")
> dev.off()
quartz
    2
>
```

Appendix E

Script for Ram Speed

E.1 Appendix: ramspeed.sh

The ramspeed.sh script

```
#! /bin/bash
timestamp=$(date +%s)
for i in 1 2 4 5 7 8 10 11
do
  COUNTER=1
  while [ $COUNTER -le 25 ]; do
      echo $i-$COUNTER | tee -a b$i-$timestamp.txt
      ./ramsmp -b$i -g 16 -m 2048 | awk '{if ( $7 ~ /^[0-9]*(\.[0-9]*)?$/ ) print $7}' |
awk '{ if ($0 !~ (/^$/) ) print $0 }' | tee -a b$i-$timestamp.txt
      let COUNTER=COUNTER+1
  done
  echo b$i-$timestamp.txt is completed
  ./datagenerator.sh b$i-$timestamp.txt;
done
echo RAM Speed test is finished
```

E.2 Appendix: datagenerator.sh

The datagenerator.sh script

```
datagenerator.sh
#! /usr/bin/perl -w
# $block = 1;
my $filename = $ARGV[0];
open (FIL, $filename) or
die "cant open file $filename\n";

my ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size, $atime, $mtime, $ctime, $blksize,
$blocks) = stat($filename);
my $newname = "ram$filename";
open (INTWRITING, ">>$newname") or die \\
"cant open file $newname\n";

$startline=1;
$skiplines=23;
$currentline=1;
while(<FIL>) {

  chomp($_);
  my $mod = $currentline % $skiplines;
  $mod = $skiplines if ($mod == 0);

```

```
  print INTWRITING $_ . "," if ($mod == $startline);

  if (tell(FIL) == $size && $startline < $skiplines) {
       print INTWRITING "\n";
       seek(FIL, 0, 0);
       $currentline = 0;
         $startline++;
  }
  ++$currentline;
}
close FIL;
close INTWRITING;
```

E.3 Appendix: ramsinglefile.sh

The ramsinglefile.sh script

```
#! /usr/bin/perl
$i = 0;
$block = 1;
my $filename = $ARGV[0];
open (FIL, "$filename") or
die "cant open file $filename\n";
while(<FIL>)
{
       $j = 0;
       my($line) = $_;
       if ($i == 0)
       {
       $i++;
       }else
       {
              open (SEP, ">>$block-$filename") or die \\
              "cant open file $block$filename\n";
              chomp($line);
              @array = split(",",$line);
              foreach( @array )
              {
                print SEP "$array[$j]\n";
                $j++;
              }
       $block = $block * 2;
       }
}
close FIL;
close SEP;
```

E.4 Appendix: R commands for Ramspeed

The R commads for Ramspeed

```
R version 2.13.1 (2011-07-08)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-apple-darwin9.8.0/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.41 (5874) i386-apple-darwin9.8.0]

[Workspace restored from /Users/naveedy/.RData]
[History restored from /Users/naveedy/.Rapp.history]

> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/1-ramb2-1333028518.txt")
> b21k=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/16-ramb2-1333028518.txt")
> b216k=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/256-ramb2-1333028518.txt")
> b2256k=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/1024-ramb2-1333028518.txt")
> b21m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/4096-ramb2-1333028518.txt")
> b24m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/16384-ramb2-1333028518.txt")
> b216m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/32768-ramb2-1333028518.txt")
> b232m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/65536-ramb2-1333028518.txt")
> b264m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/524288-ramb2-1333028518.txt")
> b2512m=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/1048576-ramb2-1333028518.txt")
> b21g=values$V1
> values=read.table("/Users/naveedy/Desktop/(b2) Integer and reading/2097152-ramb2-1333028518.txt")
> b22g=values$V1
> summary(b21k)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  37380   37660   37750   37770   37870   38100
> summary(b216k)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  37640   37840   38020   37980   38120   38160
> summary(b2256k)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  26440   26640   26910   26900   27150   27420
> summary(b21m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  25270   25390   25440   25450   25560   25600
> summary(b24m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  25240   25310   25340   25380   25460   25520
> summary(b216m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16090   16250   16300   16330   16410   16630
> summary(b232m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15510   15610   15690   15670   15730   15780
> summary(b264m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15480   15580   15660   15660   15730   15800
> summary(b2512m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15460   15630   15660   15650   15710   15740
> summary(b21g)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15530   15620   15650   15660   15720   15760
> summary(b22g)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15510   15630   15650   15670   15720   15760
> boxplot(b21k,b216k,b2256k,b21m,b24m,b216m,b232m,b264m,b2512m,b21g,b22g, outline=FALSE,
ylab="MBytes/s", xlab="Block size", names=c("1KB","16KB","256KB","1MB","4MB","16MB","32MB",
"64MB","512MB","1GB","2GB"), main="Integer and reading")
> jpeg("/Users/naveedy/Desktop/IntR.jpeg")
> boxplot(b21k,b216k,b2256k,b21m,b24m,b216m,b232m,b264m,b2512m,b21g,b22g, outline=FALSE,
ylab="MBytes/s", xlab="Block size", names=c("1KB","16KB","256KB","1MB","4MB","16MB","32MB",
"64MB","512MB","1GB","2GB"), main="Integer and reading")
> dev.off()
null device
          1
>
```